POLITECHNIKA POZNAŃSKA

Wydział Informatyki
Instytut Informatyki

Zastosowanie metod opartych na teorii grafów do rozwiązywania wybranych problemów analizy sekwencji nukleotydowych i aminokwasowych

Tomasz Głowacki

Rozprawa doktorska

Promotor:
dr hab. inż. Piotr Formanowicz, profesor PP

Poznań 2013

Serdecznie dziękuję

dr hab. inż. Piotrowi Formanowiczowi, profesorowi PP
za powierzenie ciekawego tematu badań, wyrozumiałość,
poświęcony czas i wskazówki w trakcie jego realizacji

Spis treści

Spis rysunków

Spis tablic

Skróty i oznaczenia

DNA Kwas deoksyrybonukleinowy

RNA Kwas rybonukleinowy

Rozdział 1

Wprowadzenie

1.1 Wstęp

W drugiej połowie XX wieku rozpoczął się ogromny postęp we wszystkich dziedzinach biologii molekularnej. Powstały i rozwinęły się zupełnie nowe gałęzie nauki na pograniczu matematyki, biologii oraz nieco później informatyki. Przykładami nowych, szybko rozwijających się nauk jest bioinformatyka czy biologia obliczeniowa.

W biologii obliczeniowej z powodzeniem stosuje się skomplikowane modele matematyczne, by przedstawić zachodzące w przyrodzie procesy lub wspomóc przetwarzanie ogromnej ilości informacji.

Teoria grafów jest jednym z działów matematyki często wykorzystywanych do modelowania zjawisk biologicznych. Warto by tu wspomnieć modele grafowe wykorzystywane w sekwencjonowaniu DNA [8], modelowaniu drzew filogenetycznych [9], czy przewidywaniu funkcji białek [15].

Autor niniejszej rozprawy skupia się na analizie sekwencji białek i DNA oraz proponuje własne modele grafowe do rozwiązania zagadnień związanych z sekwencjami: rekonstrukcji sekwencji białkowych na podstawie eksperymentów chemicznych oraz projektowania bibliotek sekwencji DNA o minimalnej wzajemnej zdolności do hybrydyzacji.

Jednym z ważniejszych problemów biologii obliczeniowej jest ustalanie budowy peptydów [23]. Peptydy są to wielocząsteczkowe związki składające się z aminokwasów połączonych w długie sekwencje. Długie peptydy, których sekwencje składają się z więcej niż 100 aminokwasów, nazywane są białkami [45].

Znajomość budowy białek pozwala na przewidywanie ich funkcji w organizmie, a co za tym idzie, może mocno wesprzeć przemysł farmaceutyczny w projektowaniu leków. Pierwszym krokiem w celu określenia budowy białek jest rozpoznanie sekwencji aminokwasów w cząsteczce.

Brak bezpośrednich metod chemicznych do rozpoznania długich sekwencji aminokwasowych rodzi naturalną potrzebę zastosowania metod informatycznych umożliwiających odtworzenie tego rodzaju sekwencji. Odczytywanie krótkich łańcuchów peptydowych nazywamy sekwencjonowaniem.

Współczesne metody chemii analitycznej pozwalają na określenie sekwencji peptydów nieprzekraczających 50 aminokwasów (metoda Edmana) [45]. Szybko rozwijającą się metodą identyfikacji związków chemicznych, w tym ustalania sekwencji białek, jest spektrometria masowa. Wynikiem działania spektrometru jest widmo masowe przedstawiące statystyczny rozkład jonów o różnej masie w oryginalnej cząsteczce [32]. Jednym z podejść do identyfikacji białek na podstawie widma masowego jest przeszukiwanie baz danych w poszukiwaniu tego widma. Takie podejście pozwala na rozpoznać jedynie znane białko. Metody określania budowy białka bazujące jedynie na widmie masowym tj. bez

odwoływania się do baz danych określane są jako metody de novo. W niniejszej rozprawie analizuje się jeden z problemów sekwencjonowania i proponuje wielomianowy algorytm do jego rozwiązania.

Jeden z rozdziałów niniejszej rozprawy poświęcony jest zagadnieniu konstruowania bibliotek oligonukleotydów antykomplementarnych. Są to biblioteki łańcuchów DNA, których globalna zdolność do hybrydyzacji między sobą jest minimalna. Istnieje kilka zastosowań tego typu bibliotek. Jednym z nich są klasyczne komputery DNA, które zostały zaproponowane w 1994 roku przez Adlemana [3]. Cząsteczki DNA kodują pewne informacje, a obliczenia zachodzą dzięki reakcjom chemicznym, w których te cząsteczki uczestniczą. Adleman zaproponował rozwiązanie problemu szukania ścieżki Hamiltona w grafie skierowanym. W doświadczeniu zaproponowanym przez niego część łańcuchów koduje wierzchołki grafu, pozostała część koduje łuki między wierzchołkami. Hybrydyzacja trzech łańcuchów odpowiadającym wierzchołkowi łukowi i kolejnemu wierzchołkowi symbolizuje przejście w grafie z pierwszego wierzchołka do drugiego wybranym łukiem. Obliczenia toczą się równolegle, w tym znaczeniu idea komputerów DNA przypomina do pewnego stopnia ideę Niedeterministycznej Maszyny Turinga. Aby zwiększyć wydajność reakcji zachodzących podczas obliczeń komputerów DNA, powinno dążyć się do minimalizacji prawdopodobieństwa hybrydyzacji cząsteczek, których połączenie nie koduje powstania cząsteczki kodującej rozwiązanie danego problemu. W tym celu tworzy się biblioteki oligonukleotydów antykomplementarnych. Elementy tej biblioteki umożliwiają kodowanie instancji za pomocą oligonukleotydów w taki sposób, aby minimalizować prawdopodobieństwo niechcianych hybrydyzacji.

Niniejsza rozprawa doktorska składa się z ośmiu rozdziałów, których zawartość jest następująca:

- Rozdział I zawiera uzasadnienie podjęcia badań oraz wstępne założenia pracy na podstawie których sformułowano tezę.

- Rozdział II poświęcony jest podstawom matematycznym oraz informatycznym niezbędnym do zrozumienia dalszych rozważań przedstawionych w rozprawie.

- Rozdział III poświęcony jest wybranym zagadnieniom biologicznym. Opisano w nim peptydy oraz DNA, gdyż praca skupia się na rozwiązywaniu kilku problemów analizy sekwencji tych dwóch typów cząsteczek. Przedstawiono biologiczną motywację do podjęcia badań.

- Rozdział IV dotyczy problemu budowania bibliotek oligonukleotydów antykomplementarnych. W rozdziale tym zdefiniowano problem i zaproponowano algorytm do jego rozwiązania. Rozdział ten jest również krótkim wprowadzeniem do klasycznych komputerów DNA, gdyż zaproponowane metody znajdują zastosowanie przy projektowaniu algorytmów przeznaczonych do wykonania za pomocą tego typu komputerów.

- Rozdział V dotyczy ustalania sekwencji krótkich łańcuchów peptydowych (sekwencjonowania). W rozdziale tym opisano dwa popularne podejścia do sekwencjonowania: degradację Edmana oraz spektrometrię masową. Przedstawiono krótką klasyfikację problemów drugiego typu. Zaproponowano algorytm sekwencjonowania de novo do rozwiązania problemu sekwencjonowania opartego na danych pochodzących z rzeczywistego eksperymentu spektrometrycznego.

- Rozdział VI dotyczy metod asemblacji krótkich łańcuchów peptydowych. Opisano eksperyment biochemiczny, którego wyniki posłużyły do zdefiniowania problemów

kombinatorycznych będących modelami problemów asemblacji. Przedstawiono klasyfikację problemów asemblacji oraz zaproponowano wielomianowy dokładny algorytm do rozwiązania jednego z nich. W pracy pokazano, że przypadek asemblacji bez błędów jest łatwy obliczeniowo. Podano formalną definicję grafów peptydowych będących reprezentacją tego problemu. Dla problemu asemblacji trudnego obliczeniowo zaproponowano metodę GRASP oraz algorytm ewolucyjny. Przedstawiono również heurystykę dedykowaną.

- Rozdział VII przedstawia wyniki eksperymentu obliczeniowego przeprowadzonego dla NP-trudnego problemu asemblacji. Metody zaproponowane w rozdziale VI (algorytm ewolucyjny, GRASP oraz heurystyka dedykowana) zostały zaimplementowane i przetestowane. Wyniki zostały przedstawione w postaci tabel oraz wykresów.

- Rozdział VIII jest podsumowaniem rozprawy. Wskazano nowe, potencjalne kierunki badań, które pojawiły się podczas pisania tej pracy.

1.2 Uzasadnienie podjęcia badań

W piątym i szóstym rozdziale niniejszej rozprawy autor koncentruje się na określeniu sekwencji białkowej. Proponowana jest metoda sekwencjonowania de novo oraz algorytmy asemblacji pozwalające na odtworzenie długich łańcuchów peptydowych na podstawie krótszych, pochodzących z fazy sekwencjonowania. Rozważane są różnego rodzaju błędy pochodzące z fazy eksperymentu chemicznego, dzięki czemu analizowane metody lepiej opisują rzeczywistość.

Znajomość sekwencji białkowych jest pierwszym krokiem do poznania ich budowy przestrzennej oraz ich własności, co jest jednym z kluczowych zagadnień współczesnej biologii obliczeniowej. Analiza nowych eksperymentów chemicznych, takich jak asemblacja przy wykorzystaniu endopeptydaz oraz sekwencjonowanie za pomocą spektrometrii mas pociąga za sobą konieczność zdefiniowania nowych problemów kombinatorycznych, które należy rozważyć w fazie obliczeniowej oraz konstrukcji algorytmów rozwiązujących te problemy.

Motywacją do podjęcia badań dotyczących bibliotek oligonukleotydów antykomplementarnych jest stworzenie wielomianowej metody, która pozwoli zaproponować lepsze kodowanie instancji problemów dla komputerów DNA.

Bazując na powyższych wytycznych określono założenia wstępne pracy:

- Zaprezentowane modele i metody asemblacji oraz sekwencjonowania łańcuchów peptydowych pozwolą na odtworzenie łańcuchów białkowych na podstawie przeprowadzonych eksperymentów chemicznych.

- Metoda sekwencjonowania de novo pozwoli na poznanie nowych, nieznanych sekwencji aminokwasowych.

- Metoda tworzenia bibliotek oligonukleotydów antykomplementarnych pozwoli na budowanie zbiorów oligonukleotydów, których zdolność do wzajemnej hybrydyzacji jest minimalna.

- Możliwość adaptacji wielu modeli termodynamicznych do metody tworzenia bibliotek oligonukleotydów antykomplementarnych pozwoli na łatwe dostosowanie tej metody do konkretnych wymagań.

- Ustalenie sekwencji peptydowych pozwoli na dalsze badania dotyczące przestrzennej budowy peptydów oraz ich funkcjonalności.

Postawiona na podstawie powyższych założeń teza brzmi:

Metoda sekwencjonowania peptydów de novo oraz metody asemblacji długich łańcuchów peptydowych pozwolą na kompleksowe określenie budowy pierwszorzędowej dla długich łańcuchów peptydowych (białek). Zaproponowana w pracy metoda budowania bibliotek oligonukleotydów antykomplementarnych przyczyni się do poprawienia efektywności algorytmów projektowanych dla klasycznych komputerów DNA.

Jak wspomniano, badania skupiają się na wybranych zagadnieniach dotyczących analizy sekwencji: odtwarzaniu sekwencji peptydowych na podstawie danych pochodzących z eksperymentu chemicznego oraz projektowaniu sekwencji DNA o minimalnej wzajemnej zdolności do hybrydyzacji. Celem rozprawy jest wykorzystanie teorii grafów do zamodelowania i rozwiązania tych problemów. Zaproponowano nowe modele grafowe oraz algorytmy oparte na teorii grafów.

Główne cele i zadania niniejszej pracy doktorskiej są określone następująco:

- Opracowanie modeli grafowych dla problemów asemblacji peptydów dla przypadku bez błędów oraz w przypadku z błędami pochodzącymi z fazy chemicznej.

- Zdefiniowanie grafów będących reprezentacją problemu asemblacji bez błędów. Zaprojektowanie metody rozpoznawania tych grafów oraz zbadanie ich własności.

- Zbadanie problemów kombinatorycznych asemblacji peptydów w tym ich złożoności obliczeniowej, w podstawowej wersji bez błędów, jak i z uwzględnieniem błędów pochodzących z eksperymentu chemicznego, ze szczególnym uwzględnieniem błędów pochodzących z fazy trawienia peptydu.

- Opracowanie metaheurystyk dla trudnego przypadku asemblacji peptydów oraz przeprowadzenie eksperymentu obliczeniowego na sztucznie stworzonych oraz na rzeczywistych sekwencjach peptydowych.

- Opracowanie metody de novo sekwencjonowania peptydów za pomocą spektrometru masowego. Metoda de novo pozwala na ustalenie sekwencji aminokwasów bazując na widmie masowym, może więc być wykorzystana do rozpoznawania nowych peptydów.

- Opracowanie wielomianowego algorytmu opartego na teorii grafów do tworzenia bibliotek oligonukleotydów antykomplementarnych, który pozwoli na adaptację różnych modeli opisujących zdolność hybrydyzacji łańcuchów DNA. Biblioteki te mogą posłużyć do optymalizacji algorytmów przeznaczonych do wykonania za pomocą komputerów DNA.

- Klasyfikacja problemów asemblacji z uwzględnieniem różnego rodzaju informacji oraz błędów pochodzących z eksperymentu chemicznego. Klasyfikacja pozwoli na lepsze zrozumienie wpływu błędów pochodzących z eksperymentu chemicznego na złożoność obliczeniową tych problemów.

Rozdział 2

Podstawy matematyczne i informatyczne

2.1 Wprowadzenie

Niniejszy rozdział stanowi wprowadzenie do zagadnień matematycznych niezbędnych do zrozumienia rozważań przedstawionych w pozostałych rozdziałach tej rozprawy. Poniżej przedstawiono podstawy matematyczne dotyczące:

- zbiorów
- języków formalnych
- złożoności obliczeniowej problemów kombinatorycznych
- programowania dynamicznego
- teorii grafów

Rozdział ten zawiera również wprowadzenie do metaheurystyk i programowania dynamicznego.

Omówione szczegółowo w niniejszym rozdziale zagadnienia to m. in.:

- algorytm znajdujący obwód Eulera, który posłużył do zaprojektowania algorytmu konstruującego bibliotekę oligonukleotydów antykomplementarnych (rozdział 4);
- definicja grafów de Bruijna - grafy te posłużyły do zdefiniowania modelu problemu budowy bibliotek (rozdział 4);
- algorytm Dijkstry znajdujący najkrótszą ścieżkę w grafie, który zostanie wykorzystany do rozwiązania problemu sekwencjonowania de novo (rozdział 5);
- definicja problemu plecakowego, który zostanie wykorzystany do interpretacji widma ze spektrometru masowego (rozdział 5);
- definicje metaheurytyki GRASP oraz algorytmu ewolucyjnego; metody te zostaną wykorzystane do rozwiązania problemów asemblacji długich peptydów (rozdział 6);
- definicje grafów sprzężonych, indukowanych, rozpinających, dwudzielnych; wykorzystano je do skonstruowania modeli asemblacji (rozdział 6);

- definicje alfabetu, języka, słowa; definicje będą pomocne do opisania założeń dotyczących etykiet wierzchołków w modelach asemblacji (rozdział 6) ;
- definicja problemu kolorowania grafów. W rozprawie wykorzystano problem kolorowania grafu dwoma kolorami do opracowania algorytmu rozpoznającego grafy peptydowe (rozdział 6); warto tutaj wspomnieć, że model grafów peptydowych dla problemu z błędami (rozdział 6) jest oparty na przyznawaniu wierzchołkom kolorów; podejście to nie bazuje jednak na klasycznym kolorowaniu wierzchołkowym.

2.2 Wprowadzenie do teorii mnogości

Pojęcie **zbioru** oraz pojęcie **należenia do zbioru** są pojęciami pierwotnymi teorii mnogości, czyli nie definiuje się ich językiem teorii, tylko podaje definicję znaczeniową.

Intuicyjnie - zbiorem można nazwać nieuporządkowany zestaw elementów bez wyróżnionej kolejności.

Jeśli pewien obiekt a należy do zbioru A, to zapisuje się to jako:

$$a \in A$$

Jeśli obiekt a nie należy do zbioru A to zapisuje się to:

$$a \notin A$$

Określenie pewnego zbioru A jest możliwe na dwa sposoby tzn. poprzez podanie elementów tego zbioru, np. $A = \{2, 3, 4, 5\}$ lub poprzez zdefiniowanie warunku, który spełniają wszystkie elementy tego zbioru (i żaden element spoza zbioru), np. $A = \{x : x \in NN \wedge x \geqslant 2 \wedge x \leqslant 5\}$.

Poniżej zdefiniowano podstawowe **działania na zbiorach** [42]:

- Suma zbiorów $(A \cup B)$ to zbiór, do którego należą wszystkie elementy zbioru A oraz wszystkie elementy zbioru B i żaden inny element: $a \in (A \cup B) \Leftrightarrow [(a \in A) \vee (a \in B)]$.
- Iloczyn zbiorów $(A \cap B)$ to zbiór, do którego należą wszystkie elementy, które jednocześnie należą do zbioru A oraz do zbioru B i żaden inny element: $a \in (A \cap B) \Leftrightarrow [(a \in A) \wedge (a \in B)]$.
- Różnica zbiorów $(A \backslash B)$ to zbiór, do którego należą elementy zbioru A, które jednocześnie nie należą do zbioru B i żaden inny element: $a \in (A \backslash B) \Leftrightarrow [(a \in A) \wedge (a \notin B)]$.
- Różnica symetryczna zbiorów $(A \oplus B)$ to zbiór, do którego należą te wszystkie te i tylko te elementy, które należą do zbioru A lub do zbioru B, ale nie należą do obydwu zbiorów naraz.

Poniżej zdefiniowano pojęcie **inkluzji** (oznaczanej symbolem $\subseteq$), czyli zawierania się zbioru w innym zbiorze [42]:

$$A \subseteq B \Leftrightarrow [(a \in A) \Rightarrow (a \in B)]$$

Zbiór A zawiera się w zbiorze B, gdy każdy element zbioru A należy do zbioru B. Zbiór A nazywany jest podzbiorem zbioru B, natomiast zbiór B nadzbiorem zbioru A.

Jeśli wszystkie rozważane zbiory są podzbiorami pewnego ustalonego zbioru U, to zbiór U nazywany jest przestrzenią.

Dopełnieniem zbioru A do zbioru U nazywany jest zbiór $\overline{A} = U \backslash A$.

Zbiór, który nie posiada żadnego elementu nazywany jest zbiorem pustym i oznaczany jest symbolem $\emptyset$. Przykładowo, jeśli zbiór A jest zbiorem pustym, to można to zapisać jako $A = \emptyset$ [42].

2.3 Elementy teorii języków formalnych

W celu zdefiniowania języka formalnego przytoczone zostaną najpierw definicje alfabetu oraz słowa [42].

Alfabet to skończony niepusty zbiór Σ zawierający symbole nazywane literami alfabetu Σ.

Słowem nazywa się dowolny skończony ciąg liter ze zbioru Σ.

Zbiór wszystkich słów zbudowanych z liter alfabetu Σ oznaczany jest przez Σ^*.

Dowolny podzbiór zbioru Σ^* nazywany jest **językiem** nad alfabetem Σ.

Język $L(G)$ jest zbiorem wszystkich słów, które można wyprowadzić z gramatyki G.

2.4 Podstawy teorii złożoności obliczeniowej

Problem obliczeniowy jest opisany za pomocą skończonego zbioru parametrów wejściowych oraz warunków, jakie spełnia wynik. Problemy kombinatoryczne można podzielić na:

- decyzyjne
- optymalizacyjne
- przeszukiwania

Problemem decyzyjnym π jest skończony zbiór parametrów oraz postawione pytanie, na które odpowiedź brzmi "TAK" albo "NIE".

Po ustaleniu wartości wszystkich parametrów problemu decyzyjnego otrzymuje się konkretny problem I, nazywany również instancją. Problem decyzyjny π można również zdefiniować jako zbiór D_π wszystkich konkretnych problemów oraz jego podzbiór $Y_p i \subseteq D_\pi$, gdzie Y_π zawiera wszystkie (i tylko te) konkretne problemy, dla których odpowiedź brzmi "tak"[13].

Problem optymalizacyjny to problem wymagający minimalizacji albo maksymalizacji pewnej funkcji celu (ekstremalizacji).

Problem przeszukiwania π to zbiór konkretnych problemów D_π oraz zbiór rozwiązań $Z_\pi(I)$ zdefiniowany dla każdego $I \in D_\pi$ [13]. Rozwiązaniem konkretnego problemu przeszukiwania dla $I \in D_\pi$ jest dowolne rozwiązanie należące do zbioru $Z_\pi(I)$ albo odpowiedź brzmi "NIE", jeśli $Z_\pi(I)$ jest pusty.

Algorytm to skończony, uporządkowany ciąg zrozumiale zdefiniowanych czynności, niezbędnych do wykonania pewnego rodzaju zadań - rozwiązania problemu [33].

Ważnym zagadnieniem jest złożoność obliczeniowa algorytmów.

Funkcja złożoności obliczeniowej algorytmu A rozwiązującego problem π to funkcja przyporządkowująca każdej wartości rozmiaru konkretnego problemu $I \in D_\pi$ maksymalną liczbę elementarnych kroków maszyny cyfrowej potrzebnych do rozwiązania konkretnego problemu za pomocą algorytmu A. Do zdefiniowania tej funkcji koniecznie jest określenie reguły kodowania i modelu obliczeń [5].

Reguła kodowania to sposób przedstawienia konkretnego problemu $I \in D_\pi$ za pomocą skończonego łańcucha $x(I)$ symboli należących do określonego alfabetu Σ. Rozmiarem konkretnego problemu nazywana jest długość łańcucha $x(I)$.

Na potrzeby dalszych rozważań zostanie wprowadzona notacja O: funkcja $f(k)$ jest rzędu $g(n)$, co oznaczane jest jako $O(g(n))$, gdy istnieją takie stałe n_0 oraz c, że $|f(n)| <= c|g(n)|$ dla $n > n_0$ [5].

Jeśli złożoność obliczeniowa pewnego algorytmu jest $O(p(n))$, gdzie p jest wielomianem a n rozmiarem rozwiązywanego konkretnego problemu, to algorytm nazywany jest algorytmem wielomianowym. Jeśli takie ograniczenie nie istnieje, to algorytm nazywany jest algorytmem wykładniczym.

Zdefiniowane zostaną dwa abstrakcyjne modele obliczeń: deterministyczna maszyna Turinga (DTM) ora niedeterministyczna maszyna Turinga (NDTM). Definicje te zostaną następnie wykorzystane do zdefiniowania klas złożoności problemów.

Deterministyczną maszynę Turinga można przedstawić jako skończony zbiór stanów Q, wyróżniony stan początkowy $Q_0 \in Q$, zbiór stanów końcowych F, skończony zbiór dopuszczalnych symboli Γ, symbol pusty $B \in \Gamma$, zbiór symboli wejściowych $\Sigma = \Gamma$ B oraz funkcję przejść δ, którą definiuje się w następujący sposób: $\delta : \Gamma \times Q \rightarrow Q \times \Gamma \times \{L, P, -\}$. DTM składa się z mechanizmu sterującego, taśmy zawierającej nieskończoną liczbę komórek (ponumerowanych liczbami całkowitymi), oraz głowicy odczytująco-zapisującej. Dane wejściowe programu (dla funkcji przejść) stanowi łańcuch symboli ze skończonego zbioru symboli Σ. Każdy symbol zapisany jest w kolejnej komórce taśmy, rozpoczynając od komórki 1. Na początku głowica znajduje się nad komórką 1. Pojedynczy krok programu to odczytanie zawartości bieżącej komórki i zapisanie w komórce nowej wartości oraz zmiana stanu, zgodnie z funkcją przejść. Zgodnie z informacjami zapisanymi w tej funkcji, może również nastąpić przesunięcie głowicy o jedną komórkę w lewo lub jedną w prawo. Następuje ponowny odczyt komórki przez głowicę. Działanie programu kończy się po osiągnięciu jednego z wyróżnionych oznaczających odpowiednio odpowiedzi "TAK" lub "NIE", czyli znalezieniu rozwiązania problemu. Jeśli dla danego problemu decyzyjnego można zdefiniować Deterministyczną Maszynę Turinga, która rozwiązuje ten problem, to można go również rozwiązać za pomocą rzeczywistej maszyny obliczeniowej (komputera).

Niedeterministyczna Maszyna Turinga (NDTM) [38] to DTM wyposażona w dodatkowy moduł generujący. Program dla NDTM jest określany w taki sam sposób jak dla DTM. Wykonanie programu różni się jednak, gdyż składa się z dwóch etapów. W pierwszym etapie dla łańcucha $x(I)$, który koduje dane konkretnego problemu I, następuje wygenerowanie łańcucha S zbudowanego z symboli Γ. Łańcuch ten jest zapisywany na taśmie. Drugi etap jest analogiczny do wykonywania programu na DTM, czyli sprawdza się, czy S spełnia warunki określone w pytaniu problemu decyzyjnego π - czyli czy stan końcowy działania maszyny oznacza odpowiedź "TAK" czy odpowiedź "NIE". NDTM rozwiązuje problem decyzyjny π, gdy dla każdego $I \in \pi$:

- Jeśli odpowiedź dla I brzmi "TAK", to zostanie wygenerowany łańcuch S, który wraz z $x(I)$ spowoduje, że po zakończeniu programu przez NDTM maszyna osiągnie stan końcowy odpowiadający odpowiedzi "TAK"

- Jeśli odpowiedź brzmi "NIE", to dla każdego wygenerowanego łańcucha S NDTM zakończy działanie w stanie odpowiadającym odpowiedzi "NIE" lub sprawdzanie nie zakończy się w skończonym czasie.

Problem decyzyjny należy do klasy P, gdy istnieje deterministyczna maszyna Turinga, która rozwiązuje ten problem w co najwyżej wielomianowym czasie [37].

Problem decyzyjny należy do klasy NP, gdy istnieje niedeterministyczna maszyna Turinga, która rozwiązuje go w czasie co najwyżej wielomianowym [37].

Jak dotąd nie podano formalnego dowodu, czy w klasie NP istnieją inne problemy niż w klasie P (nie udowodniono, czy P $\neq$ NP).

Można powiedzieć, że klasa P zawiera te problemy, dla których można zaproponować algorytm wielomianowy. Do klasy NP należą te problemy, dla których w czasie wielomianowym można zweryfikować poprawność znanego rozwiązania.

Poniżej zdefiniowana zostanie podklasa klasy NP, nazywana podklasą problemów NP-zupełnych. W tym celu wprowadzona zostanie wpierw definicja transformacji wielomianowej [5].

Transformacją wielomianową problemu π_2 do problemu π_1 ($\pi_2 \propto \pi_1$) nazywamy funkcję $f : D_{\pi_2} \to D_{\pi_1}$, która spełnia następujące warunki:

1) dla każdego $I_2 \in D_{\pi_2}$, odpowiedź brzmi "TAK" wtedy i tylko wtedy, gdy dla $f(I_2)$ odpowiedź brzmi również "TAK";

2) czas obliczania funkcji f przez DTM dla każdego $I_2 \in D_{\pi_2}$ jest ograniczony od góry przez wielomian od $N(I_2)$.

Problem decyzyjny π_1 jest NP-zupełny jeżeli $\pi_1 \in$ NP i dla każdego innego problemu decyzyjnego $\pi_2 \in$ NP, $\pi_2 \propto \pi_1$.

Z powyższej definicji wynika, że wszystkie problemy klasy problemów NP-zupełnych wielomianowo transformują się do siebie nawzajem. Wynika z tego, że aby wykazać NP-zupełność badanego problemu, trzeba pokazać wielomianową transformację dowolnego problemu NP-zupełnego do niego.

Przy założeniu, że $P \neq$ NP problemy należące do klasy problemów NP-zupełnych można rozwiązać przy użyciu algorytmów wykładniczych.

Przez $N(I)$ oznaczony zostanie rozmiar instancji I a przez $Max(I)$ maksymalna liczba występująca w tej instancji.

Problem decyzyjny π jest problemem liczbowym, gdy nie istnieje wielomian p, taki że $Max(I) \leqslant p(N(I))$ dla każdego $I \in D_\pi$.

Dla niektórych liczbowych problemów decyzyjnych można skonstruować algorytmy pseudowielomianowe. Są to algorytmy, których funkcja złożoności obliczeniowej jest ograniczona z góry przez wielomian od $N(I)$ oraz przez $Max(I)$.

Problem nieliczbowy π_p jest takim podproblemem π, który zawiera tylko te instancje problemu π, dla których istnieje wielomian p, taki że $Max(I) \leqslant p(N(I))$.

Problem decyzyjny π jest **silnie NP-zupełny**, gdy należy do klasy NP oraz istnieje pewien wielomian p zdefiniowany dla liczb całkowitych, taki że π_p jest NP-zupełny [5].

Z powyższych definicji wynika, że NP-zupełny problem nieliczbowy jest silnie NP-zupełny.

Aby wykazać silną NP-zupełność liczbowego problemu decyzyjnego π należy znaleźć wielomian p, dla którego π_p jest NP-zupełny. Można jednak postąpić inaczej i skorzystać z transformacji pseudowielomianowej:

Pseudowielomianową transformacją [5] problemu π_2 do problemu π_1 nazywa się funkcję $f : D_{\pi_2} \to D_{\pi_1}$, taką że:

1) dla każdej instancji problemu $I_2 \in D_{\pi_2}$, odpowiedź brzmi "TAK" wtedy i tylko wtedy, gdy dla $f(I_2)$ brzmi także "TAK";

2) funkcja f może być obliczona (przez DTM) w czasie ograniczonym od góry przez wielomian zależny $Max_2(I_2)$ i $N_2(I_2)$;

3) istnieje wielomian q_1, taki że dla każdego $I_2 \in D_{\pi_2}$ $q_1(N_1(f(I_2))) \geqslant N_2(I_2)$;

4) istnieje wielomian dwóch zmiennych q_2, taki że dla każdego $I_2 \in D_{\pi_2}$, zachodzi $Max_1(f(I_2)) \leqslant q_2(Max_2(I_2), N_2(I_2))$;

Problem $\pi_1 \in$ NP jest silnie NP-zupełny, gdy istnieje pseudowielomianowa transformacja znanego problemu silnie NP-zupełnego π_2 do π_1.

Powyższa definicja pozwala na wykazanie silnej NP-zupełności bez znajdowania podproblemu π_p.

Warto tutaj wspomnieć, że porównując pseudowielomianiwą transformację i wielomianową transformację, to w przypadku tej pierwszej osłabiony został warunek wielomianowego czasu realizacji algorytmu realizującego funkcję f. Dodatkową różnicą jest fakt, że w przypadku pseudowielomianowej transformacji zakłada się, że rozmiar instancji nie może zmaleć wykładniczo a największa wartość liczbowa w instancji nie może wzrosnąć wykładniczo.

Zostanie zdefiniowana **wielomianowa transformacja Turinga** [5] problemu przeszukiwania π_1 do problemu przeszukiwania π_2 oznaczana jako $\pi_1 \propto_T \pi_2$. Jest to algorytm A rozwiązujący problem π_1 z zastosowaniem hipotetycznej procedury P rozwiązującej π_2. Czas wykonania algorytmu A przez DTM jest wielomianowy, jeśli procedura P może być wykonana przez DTM w czasie wielomianowym.

Problem przeszukiwania π_2 jest NP-trudny, jeśli istnieje NP-zupełny problem pi_1, taki że $\pi_1 \propto_T \pi_2$ [5]

Poniżej omówiono zależności między problemami decyzyjnymi, optymalizacyjnymi i przeszukiwania.

Istnieje związek pomiędzy problemami przeszukiwania a problemami optymalizacyjnymi. Algorytm rozwiązuje problem przeszukiwania, gdy znajduje dowolne rozwiązanie ze zbioru rozwiązań. Problem optymalizacyjny można przedstawić jako problem przeszukiwania, gdzie zbiór rozwiązań zawiera wszystkie rozwiązania, które ekstremalizują wartość funkcji celu.

Z każdym problemem optymalizacyjnym można powiązać problem decyzyjny. W wersji optymalizacyjnej szukane jest rozwiązanie problemu o minimalnej (albo maksymalnej) wartości funkcji celu. W wersji decyzyjnej problemu zadane jest pytanie o istnienie rozwiązania, dla którego wartość funkcji celu spełnia pewną nierówność. Widać zatem, że problem decyzyjny nie jest obliczeniowo trudniejszy od odpowiadającego mu problemu optymalizacyjnego.

Wiadomo, że jeśli problem decyzyjny jest NP-zupełny to odpowiadający mu problem przeszukiwania jest NP-trudny. Podobnie, jeśli problem decyzyjny jest silnie NP-zupełny, to odpowiadający mu problem przeszukiwania jest silnie NP-trudny [37].

Na zakończenie tego podrozdziału warto dodać, iż aby udowodnić, że problem jest łatwy obliczeniowo należy pokazać wielomianowy algorytm, który go rozwiązuje. Zostało to podkreślone, gdyż rozprawie zostanie przedstawiony taki dowód dla jednego z problemów asemblacji.

2.5 Wybrane zagadnienia teorii grafów

2.5.1 Wprowadzenie

Teoria grafów to dział matematyki zajmujący się badaniem własności grafów. Za pierwszą pracę w tej dziedzinie uważa się pracę Leonharda Eulera z 1736 roku dotyczącą zagadnienia mostów królewieckich [25]. Problem, który rozważał Euler jest następujący: czy można przejść przez wszystkie siedem mostów znajdujących się na rzece Przegoła w Królewcu w ten sposób, aby przez każdy most przejść tylko raz i zakończyć podróż w tym samym miejscu, gdzie została rozpoczęta.

Grafy pozwalają na przedstawienie informacji w zorganizowany sposób. Mogą zostać wykorzystane do modelowania informacji geograficznej: opisu map, planowania optymalnych dróg, obliczania odległości. Teoria grafów znajduje również swoje zastosowanie w wielu innych dziedzinach, takich jak fizyka, analiza i projektowanie oddziaływań społecznych czy bioinformatyka. Zdefiniowane w tym rozdziale podstawowe pojęcia z teorii grafów można znaleźć w pracy [12].

2.5.2 Definicje

Na początku zdefiniowane zostaną dwa typy grafów:

Def. 2.5.1. Graf skierowany
Niech V będzie niepustym zbiorem, natomiast $A \subseteq V \times V$. Parę $G = (V, A)$ nazywamy grafem skierowanym lub digrafem, przy czym V jest zbiorem wierzchołków natomiast A jest zbiorem łuków.

Def. 2.5.2. Graf nieskierowany
Niech V będzie niepustym zbiorem, natomiast $E \subseteq \{\{v_i, v_j\} : v_i \in V \wedge v_j \in V\}$. Parę $G = (V, E)$ nazywamy grafem nieskierowanym, gdzie V jest zbiorem wierzchołków natomiast E jest zbiorem krawędzi.

Poniżej wprowadzono pojęcie incydencji:

Def. 2.5.3. Incydentność łuków i krawędzi
W przypadku grafu skierowanego łuk $(v_i, v_j) \in A$ jest incydentny z wierzchołkiem v_i oraz wierzchołkiem v_j. Analogicznie w przypadku grafu nieskierowanego krawędź $v_i, v_j \in E$ jest incydentna z wierzchołkami v_i i v_j.

W kontekście analizy grafów często rozważa się pewne sekwencje wierzchołków i krawędzi. Poniżej zdefiniowano najbardziej podstawowe sekwencje dla grafów nieskierowanych:

Def. 2.5.4. Łańcuch, długość łańcucha, łańcuch zamknięty, ścieżka, droga, cykl, obwód w grafie nieskierowanym
Dany jest graf nieskierowany G = (V,E). Łańcuchem $x - y$ w grafie G jest sekwencja: $x = v_0, e_1, v_1, e_2, v_2, ..., e_n, v_n = y$ gdzie $e_i = (v_{i-1}, v_i) \in E$ dla $i = 1, 2, ..., n$. Długość łańcucha n to liczba zawartych w nim krawędzi. Jeśli $x = y$ oraz $n > 1$ to łańcuch nazywany jest zamkniętym. W przeciwnym wypadku jest to łańcuch otwarty. Jeśli żadna krawędź nie powtarza się w łańcuchu więcej niż 1 raz, to taki łańcuch nazywany jest drogą pomiędzy x i y. Zamknięty łańcuch będący drogą nazywany jest obwodem. Łańcuch otwarty, w którym nie powtarza się żaden wierzchołek więcej niż raz nazywany jest ścieżką. Zamknięta ścieżka nazywana jest cyklem.

Powyższe pojęcia można analogicznie zdefiniować dla grafów skierowanych:

Def. 2.5.5. Łańcuch, długość łańcucha, łańcuch zamknięty, ścieżka, droga, cykl, obwód w grafie skierowanym
Dany jest graf skierowany G = (V,A). Łańcuchem $x - y$ w grafie G jest sekwencja: $x = v_0, a_1, v_1, a_2, v_2, ..., a_n, v_n = y$ gdzie $a_i = (v_{i-1}, v_i) \in A$ dla $i = 1, 2, ..., n$. Długość łańcucha n to liczba zawartych w nim łuków. Jeśli $x = y$ oraz $n > 1$ to łańcuch nazywany jest zamkniętym. W przeciwnym wypadku jest to łańcuch otwarty. Jeśli żaden łuk nie powtarza się w łańcuchu więcej niż 1 raz, to taki łańcuch nazywany jest skierowaną drogą pomiędzy x i y. Zamknięty łańcuch będący skierowaną drogą nazywany jest skierowanym obwodem. Łańcuch otwarty, w którym nie powtarza się żaden wierzchołek więcej niż raz nazywany jest skierowaną ścieżką. Zamknięta skierowana ścieżka nazywana jest skierowanym cyklem.

W kontekście badań zawartych w tej rozprawie, dwa rodzaje sekwencji grafowych zasługuje na szczególne wyróżnienie: jest to cykl Hamiltona i obwód Eulera. Poniższe definicje wprowadzają oba pojęcia:

Def. 2.5.6. Obwód Eulera
Niech G = (V,E) będzie grafem nieskierowanym. Obwód Eulera w grafie G to obwód, który przechodzi przez każdą krawędź w grafie G dokładnie raz. Natomiast droga Eulera dla grafu nieskierowanego to droga przechodząca przez każdą krawędź w grafie G dokładnie raz.

Def. 2.5.7. Droga Eulera
Niech G = (V,E) będzie grafem nieskierowanym. Droga Eulera w grafie G to droga, która przechodzi przez każdą krawędź w grafie G dokładnie raz.

Analogicznie definicja dla grafu skierowanego:

Def. 2.5.8. Skierowany obwód Eulera
Niech G = (V,A) będzie skierowanym grafem. Skierowany obwodów Eulera w skierowanym grafie G to skierowany obwód, który przechodzi przez każdy łuk w grafie G dokładnie raz.

Def. 2.5.9. Skierowana droga Eulera
Niech G = (V,A) będzie skierowanym grafem. Droga Eulera dla grafu skierowanego to droga przechodząca przez wszystkie łuki w grafie G dokładnie raz.

Kolejne definicje dotyczą cyklu Hamiltona:

Def. 2.5.10. Skierowany cykl Hamiltona
Niech $G = (V, A)$ będzie skierowanym grafem. Skierowany cykl Hamiltona w grafie G to skierowany cykl, który przechodzi przez wszystkie wierzchołki grafu dokładnie raz.

Def. 2.5.11. Skierowana ścieżka Hamiltona
Niech $G = (V, A)$ będzie skierowanym grafem. Skierowana ścieżka Hamiltona to ścieżka, która przechodzi przez wszystkie wierzchołki grafu dokładnie raz.

Analogicznie dla grafu nieskierowanego można zdefiniować:

Def. 2.5.12. Cykl Hamiltona
Niech $G = (V, E)$ będzie nieskierowanym grafem. Cykl Hamiltona to cykl przechodzący przez wszystkie wierzchołki grafu dokładnie raz.

Def. 2.5.13. Ścieżka Hamiltona
Niech $G = (V, E)$ będzie nieskierowanym grafem. Ścieżka Hamiltona to ścieżka przechodząca przez wszystkie wierzchołki grafu dokładnie raz.

Znalezienie cyklu (lub ścieżki) Hamiltona jest problemem silnie NP-trudnym. Oznacza to, że przy założeniu, że P $\neq$ NP, nie istnieje algorytm wielomianowy rozwiązujący ten problem.

Poniżej, po zdefiniowaniu niezbędnych pojęć, zostanie podany wielomianowy algorytm znajdujący obwód (albo drogę) Eulera w grafie.

Na potrzeby dalszych rozważań, wprowadzone zostanie pojęcie grafu spójnego:

Def. 2.5.14. Spójność grafu nieskierowanego
Graf nieskierowany $G = (V, E)$ jest spójny, jeśli każda para wierzchołków tego grafu jest połączona ścieżką.

Def. 2.5.15. Spójność grafu skierowanego
Graf skierowany $G = (V, A)$ jest spójny, jeśli skojarzony z nim graf nieskierowany, otrzymany przez usunięcie zwrotów łuków jest spójny. Przy czym w przypadku gdy para wierzchołków jeśli $x \in V$ i $y \in V$ jest połączona dwoma łukami (x, y) oraz (y, x) to w skojarzonym grafie jest tylko jedna krawędź łącząca te wierzchołki.

Zdefiniowane zostanie pojęcie stopnia wierzchołka:

Def. 2.5.16. Stopień wierzchołka (graf nieskierowany)
Stopniem wierzchołka $deg(x)$ grafu nieskierowanego $G = (V, E)$ nazywamy liczbę incydentnych z wierzchołkiem x krawędzi.

Def. 2.5.17. Stopień wejściowy i stopień wyjściowy wierzchołka
Dla grafu skierowanego $G = (V, A)$ definiuje się stopień wejściowy $in(x)$ wierzchołka x jako liczbę łuków wchodzących do wierzchołka x, czyli liczbę łuków $(v, x) \in A$. Stopień wyjściowy $out(x)$ wierzchołka x to liczba łuków wychodzących z wierzchołka x, czyli łuków postaci $(x, v) \in A$.

Poniżej zdefiniowano warunek konieczny i wystarczający istnienie obwodu Eulera:

Twierdzenie 2.5.1. Graf nieskierowany $G = (V, E)$ zawiera skierowany obwód Eulera wtedy i tylko wtedy, gdy jest spójny i każdy wierzchołek tego grafu ma parzysty stopień. Graf G zawiera skierowaną drogę Eulera, gdy jest spójny i dokładnie dwa wierzchołki tego grafu mają stopień nieparzysty.

Twierdzenie 2.5.2. Graf skierowany $G = (V, A)$ zawiera skierowany obwód Eulera wtedy i tylko wtedy, gdy jest spójny a stopień wejściowy każdego wierzchołka jest równy stopniu wyjściowemu każdego wierzchołka: $\forall_{v \in V}\, in(v) = out(v)$. Graf G zawiera skierowaną drogę Eulera, gdy jest spójny i dokładnie dla jednego wierzchołka tego grafu $in(v) - out(v) = 1$, dla jednego wierzchołka grafu $out(v) - in(v) = 1$ a dla pozostałych $n - 2$ wierzchołków grafu $in(v) = out(v)$.

Graf $G = (V, E)$, który posiada obwód Eulera, nazywany jest grafem Eulera. Poniższy algorytm znajduje obwód Eulera w grafie Eulera G:
Algorytm

1. Wybranie dowolnego wierzchołka v grafu G jako "wierzchołka startowego".
2. Znalezienie dowolnego obwodu rozpoczynającego się wierzchołkiem startowym. Obwód ten nazywany będzie "aktualnym obwodem".
3. Zaznaczenie wszystkich krawędzi wchodzących w skład tego obwodu.
4. Jeśli wszystkie krawędzie w G zostały zaznaczone, to obwód Eulera został znaleziony; w przeciwnym wypadku należy znaleźć obwód rozpoczynający się w wierzchołku incydentnym do niezaznaczonej krawędzi.
5. Zaznaczenie wszystkich krawędzi wchodzących w skład tego obwodu i połączenie znalezionego obwodu z "aktualnym obwodem".
6. Przejście do 4. kroku algorytmu.

Algorytm można analogicznie zdefiniować dla grafów skierowanych.
Zostanie wprowadzona następująca definicja:

Def. 2.5.18. p-graf
Skierowany graf $G = (V, A)$ jest skierowanym p-grafem, jeśli dla dowolnej pary wierzchołków tego grafu (v_i, v_j) istnieje co najwyżej p łuków z v_i do v_j.

Ze względu na liczbę łuków, które są incydentne z tą samą parą wierzchołków, można wprowadzić następujący podział grafów:

1) 1-grafy, które zawierające co najwyżej 1 łuk pomiędzy dowolną parą wierzchołków;

2) multigrafy, w których zezwala się na istnienie więcej niż jednego połączenia między tą samą parą wierzchołków (p-grafy, dla których $p > 1$).

Zostanie wprowadzona definicja podgrafu, a następnie definicje podgrafu rozpinającego i podgrafu indukowanego:

Def. 2.5.19. Podgraf grafu
Dany jest graf nieskierowany $G = (V, E)$. Jeśli $V_1 \neq \emptyset$ oraz $V_1 \in V$ oraz $E_1 \in E$, gdzie każda krawędź ze zbioru E_1 jest incydentna z wierzchołkami ze zbioru V_1 to graf $G_1 = (V_1, E_1)$ jest podgrafem grafu G.

Def. 2.5.20. Podgraf rozpinający
Dany jest graf nieskierowany $G = (V, E)$. Niech $G_1 = (V_1, E_1)$ będzie podgrafem grafu G. Jeśli $V_1 = V$, to G_1 nazywany jest podgrafem rozpinającym grafu G.

Def. 2.5.21. Podgraf indukowany
Dany jest graf nieskierowany $G = (V, E)$. Jeśli $V_2 \neq \emptyset$ oraz $V_2 \in V$ oraz $E_2 \in E$ to graf $G_2 = (V_2, E_2)$ nazywany jest grafem indukowanym grafu G, gdy zbiór E_2 zawiera wszystkie krawędzie $(x, y) \in E$, dla których $x, y \in V_2$.

Dla grafu skierowanego powyższe definicje są analogiczne.

Zbiór następników wierzchołka oraz zbiór poprzedników wierzchołka w grafie nieskierowanym definiuje się następująco:

Def. 2.5.22. Zbiór następników
Niech będzie dany graf skierowany $G = (V, A)$, zbiór następników $N^+(v)$ wierzchołka v to wszystkie wierzchołki x_i takie, że istnieje łuk $(v, x_i) \in A$, czyli:
$N^+(v) = \{x_i \in V : (v, x_i) \in A\}$

Def. 2.5.23. Zbiór poprzedników
Niech będzie dany graf skierowany $G = (V, A)$, zbiór poprzedników $N^-(v)$ wierzchołka v to wszystkie wierzchołki x_i takie, że istnieje łuk $(x_i, v) \in A$, czyli:
$N^-(v) = \{x_i \in V : (x_i, v) \in A\}$

Klasa grafów to zbiór wszystkich grafów spełniających pewne warunki. Pewne własności problemu z teorii grafów, przykładowo jego złożoność obliczeniowa, mogą zależeć od klasy grafów, dla której ten problem jest rozważany. Z tego powodu analizę często zawęża się do konkretnych klas.

Klasy grafów i twierdzenia wybrane poniżej znajdują zastosowanie w modelowaniu zagadnień biologicznych opisanych w niniejszej rozprawie.

Def. 2.5.24. Graf sprzężony [12]
Graf sprzężony $H = (A, U)$ skierowanego grafu $G = (V, A)$ jest 1-grafem ze zbiorem wierzchołków A. W grafie H istnieje łuk (a_i, a_j) wtedy i tylko wtedy, gdy wierzchołek do którego wchodzi łuk a_i w grafie G jest wierzchołkiem, z którego wychodzi łuk a_j. Skierowany graf H jest grafem sprzężonym, jeśli istnieje pewien graf G, taki że H jest grafem sprzężonym G.

Def. 2.5.25. Skierowany graf liniowy [8]
Skierowany graf jest skierowanym grafem liniowym, wtedy i tylko wtedy gdy jest grafem sprzężonym pewnego 1-grafu.

Twierdzenie 2.5.3. [8] Jeśli graf H jest grafem sprzężonym skierowanego grafu G, to istnieje skierowany obwód Eulera w G wtedy i tylko wtedy, gdy istnieje skierowany cykl Hamiltona w H.

Wniosek:
Jeśli graf H jest skierowanym grafem liniowym 1-grafu G, to istnieje skierowany obwód Eulera w G wtedy i tylko wtedy, gdy istnieje skierowany cykl Hamiltona w H. Dla grafów sprzężonych istnieje zatem możliwość rozwiązania problemu cyklu Hamiltona w czasie wielomianowym. W pracy [14] pokazano także, że w czasie wielomianowym możliwe jest rozwiązanie problemu cyklu Hamiltona w nadklasie grafów sprzężonych zwanej grafami quasi-sprzężonymi.

Twierdzenie 2.5.4. 1-Graf $G = (V, A)$ jest grafem sprzężonym wtedy i tylko wtedy, gdy dla każdej pary wierzchołków $x_i, x_j \in V$ gdzie spełniona jest następująca implikacja logiczna: $N^+(x_i) \cap N^+(x_j) \neq \emptyset \Rightarrow N^+(x_i) = N^+(x_j)$.

Twierdzenie 2.5.5. [8] 1-graf $G = (V, A)$ jest skierowanym grafem liniowym wtedy i tylko wtedy,gdy dla każdej pary wierzchołków $x_i, x_j \in V$ gdzie $i \neq j$ spełniona jest następująca implikacja:

$$\Big(x_i \neq x_j \wedge N^+(x_i) \cap N^+(x_j) \neq \emptyset\Big) \Rightarrow \Big(N^+(x_i) = N^+(x_j) \wedge N^-(x_i) \cap N^-(x_j) = \emptyset\Big)$$

Poniżej przedstawiono kolejne dwie definicje klas grafów wykorzystanych w rozprawie:

Def. 2.5.26. Nieskierowany graf dwudzielny
Nieskierowany graf $G = (V, E)$ jest nazywany grafem dwudzielnym, jeśli $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$ oraz dla każdej krawędzi $v_i, v_j \in E$, $v_i \in V_1 \wedge v_j \in V_2$.

Def. 2.5.27. Nieskierowany graf p-dzielny (wielodzielny)
Nieskierowany graf $G = (V, E)$ jest nazywany grafem P-dzielnym, jeśli $V = V_1 \cup V_2 \cup ... \cup V_p$, $V_a \cap V_b = \emptyset$ dla każdego $a \neq b$ oraz dla każdej krawędzi $\{v_i, v_j\} \in E$, wierzchołek v_i należy do innego podzbioru niż wierzchołek v_j.

Analogiczne definicje dot. grafu dwudzielnego i grafu p-dzielnego można podać dla grafów skierowanych.

W teorii grafów wprowadza się pojęcie grafu ważonego:

Def. 2.5.28. Graf ważony
Graf ważony to graf, w którym każdej krawędzi lub łukowi przyporządkowano pewną wagę.

Wagi są zwykle liczbami rzeczywistymi, mogą zostać jednak ograniczone do liczb wymiernych lub całkowitych. W tej pracy proponuje się również przyporządkowanie krawędziom (łukom) wektorów liczb rzeczywistych.

Jednym z istotnych problemów dla skierowanych grafów ważonych jest problem najkrótszej skierowanej ścieżki między dwoma wyróżnionymi wierzchołkami grafu (wierzchołkiem startowym „s" oraz wierzchołkiem końcowym „t"). Najkrótsza skierowana ścieżka pomiędzy dwoma wierzchołkami to ta, której suma wag łuków (krawędzi) wchodzących w jej skład jest minimalna.

Spośród algorytmów rozwiązujących problem najkrótszej skierowanej ścieżki w grafie, zostanie przytoczony algorytm Dijkstry [22][42], gdyż był on inspiracją do rozwiązania jednego z biologicznych zagadnień rozważanych w tej pracy. Poniżej podana została definicja problemu (analogiczną definicję można podać dla grafu nieskierowanego):

Def. 2.5.29. Problem najkrótszej skierowanej ścieżki w grafie
Instancja:
Skierowany graf ważony $G = (V, A)$.
Odpowiedź:
Najkrótsza skierowana ścieżka pomiędzy wierzchołkiem „s" a wierzchołkiem „t".

Algorytm Dijkstry znajdujący najkrótszą skierowaną ścieżkę w grafie G ma następującą postać:

1. Utworzenie tablicy odległości. Odległość do źródła„s" wynosi 0. Odległość do wszystkich pozostałych wierzchołków wynosi nieskończoność. Utworzenie zbioru zawierającego nieodwiedzone wierzchołki. Zbiór ten zawiera wszystkie wierzchołki grafu. Utworzenie tablicy poprzedników. Jako poprzednik każdego wierzchołka ustawia się wartość „brak poprzednika".

2. Jako wierzchołek aktualny przyjmuje się wierzchołek „s".

3. Obliczenie tymczasowych odległości do wszystkich nieodwiedzonych bezpośrednich następników aktualnego wierzchołka. Dla każdego wierzchołka, dla którego obliczona odległość tymczasowa jest mniejsza niż dotychczas zapisana odległość, za nową odległość do tego wierzchołka przyjmuje się obliczoną odległość tymczasową.

4. Usunięcie wierzchołka aktualnego z listy nieodwiedzonych wierzchołków.

5. Jako wierzchołek aktualny przyjmuje się nieodwiedzony wierzchołek, do którego obliczona odległość tymczasowa jest najmniejsza. Jako poprzednika nowego wierzchołka aktualnego przyjmuje się poprzedni wierzchołek aktualny.

6. Jeśli zbiór nieodwiedzonych wierzchołków jest pusty, należy zakończyć algorytm. W przeciwnym wypadku jako wierzchołek aktualny ustawia się ten spośród nieodwiedzonych wierzchołków, dla którego obliczona odległość jest najmniejsza. Należy przejść do punktu 3. algorytmu.

Powyższą procedurę można przerwać, gdy wierzchołek „t" zostanie zdjęty z listy nieodwiedzonych wierzchołków. Jeśli algorytm nie zostanie wcześniej przerwany, to zakończy się, gdy lista nieodwiedzonych wierzchołków będzie pusta. Aby było możliwe obliczenie minimalnej odległości pomiędzy wierzchołkami, graf nie może zawierać cykli o ujemnej sumie wag. W takim wypadku każde przejście tym cyklem obniżałoby długość aktualnie znalezionej ścieżki.

W teorii grafów wprowadza się pojęcie etykiety. Z danym wierzchołkiem, łukiem lub krawędzią można powiązać etykietę - słowo zbudowane nad pewnym alfabetem. Poniżej podano definicję dwóch klas grafów posiadających etykiety. Wybrano Grafy de Bruijna [21], gdyż zostaną wykorzystane w rozdziale 4., oraz grafy DNA [8], jako że stanowią ciekawą reprezentację dla niektórych problemów dotyczących sekwencji biologicznych.

Def. 2.5.30. Graf de Bruijna [21]
Graf de Bruijna B(d,k) jest grafem skierowanym, którego wierzchołki odpowiadają wszystkim słowom o długości k nad pewnym alfabetem Σ rozmiaru d. Łuki w grafie B(d,k) zdefiniowane są następująco: z wierzchołka zaetykietowanego $(x_1, x_2, ..., x_k)$ prowadzi łuk do wierzchołka zaetykietowanego $(x_2, x_3, ..., x_k, \gamma)$, gdzie $\gamma \in \Sigma$.

Na podstawie definicji można stwierdzić, że graf $B(d,k)$ zawiera d^k wierzchołków. Dodatkowo, stopień wejściowy i wyjściowy każdego wierzchołka grafu $B(d,k)$ jest równy d.

Na potrzebę podania definicji grafów DNA, zostaną wprowadzone definicje pomocnicze.

Def. 2.5.31. Graf (α,k)-etykietowalny [8]
Niech $k>1,\alpha>0$ będą liczbami całkowitymi. Wtedy 1-graf $G=(V,A)$ jest grafem (α,k)-*etykietowalnym*, jeśli możliwe jest przyporządkowanie każdemu wierzchołkowi $x\in V$ etykiety $(l_1(x),l_2(x),\ldots,l_k(x))$ o długości k takiej, że:

1. $\forall i\forall x\in V: l_i(x)\in\{1,\ldots,\alpha\}$,
2. wszystkie etykiety są różne, tj. $\forall x\neq y: (l_1(x),\ldots,l_k(x))\neq(l_1(y),\ldots,l_k(y))$,
3. istnieje łuk między $(x,y)\in A$ wtedy i tylko wtedy, gdy $k-1$ końcowych liter etykiety wierzchołka x jest równych $k-1$ początkowym literom etykiety wierzchołka y, tj.:
   $(x,y)\in A\Leftrightarrow(l_2(x),\ldots,l_k(x))=(l_1(y),\ldots,l_{k-1}(y))$

Def. 2.5.32. Niech $k>1,\alpha>0$ – wtedy $\mathscr{L}_k^\alpha$ oznacza klasę 1-grafów, które są (α,k)-etykietowalne.

Def. 2.5.33. [8] Graf H jest *grafem DNA* wtedy i tylko wtedy, gdy $\exists k>1$ takie, że $H\in\mathscr{L}_k^4$.

Kolejne definicje dotyczą kolorowania grafu:

Def. 2.5.34. Kolorowanie grafu [17]
Kolorowaniem grafu nieskierowanego $G=(V,E)$ nazywamy każdą funkcję $C:V\to N$ taką, że jeśli $C(u)=C(v)$ to $(u,v)\notin E$, dla wszystkich $u,v\in V$

Zgodnie z definicją, kolorowanie grafu polega na przyporządkowaniu wszystkim wierzchołkom grafu kolorów, w taki sposób, żeby żadne dwa wierzchołki incydentne z tą samą krawędzią nie miały tego samego koloru.

Liczba chromatyczna grafu to minimalna liczba kolorów niezbędna do jego pokolorowania. Problem znajdowania liczby chromatycznej jest silnie NP-trudny.

Warto zaznaczyć, że jeśli liczba chromatyczna grafu wynosi 2, to kolorowanie takiego grafu można znaleźć w czasie wielomianowym.

2.6 Programowanie dynamiczne

Programowanie dynamiczne [11] jest jedną z technik rozwiązywania problemów optymalizacji.

Podejście to można wykorzystać, jeśli problem posiada własność optymalnej podstruktury. Własność ta oznacza, że optymalne rozwiązanie problemu jest funkcją optymalnych rozwiązań podproblemów, a co za tym idzie funkcję celu można opisać równaniem rekurencyjnym. Rozwiązując problem za pomocą programowania dynamicznego, każdy podproblem wystarczy rozwiązać tylko raz.

Programowanie dynamiczne można wykorzystać do rozwiązania niektórych liczbowych problemów NP-zupełnych.

Funkcja złożoności obliczeniowej algorytmów opartych na programowaniu dynamicznym jest wielomianem zależnym od co najmniej dwóch zmiennych - rozmiaru problemu

$N(I)$ oraz maksymalnej wartości pewnego parametru problemu $Max(I)$. Jeśli nie istnieje wielomian p, taki że $Max(I) \leqslant p(N(I)$, to algorytm programowania dynamicznego ma złożoność pseudowielomianową.

Poniżej zdefiniowano problem plecakowy, który można rozwiązać stosując programowanie dynamiczne [5]:

Instancja:
Pojemność plecaka b, zbiór elementów $A = \{a_1, a_2, ..., a_n\}$, rozmiar $s(a_i)$ każdego elementu $a_i \in A$, wartość $w(a_i)$ każdego elementu $a_i \in A$

Odpowiedź:
Podzbiór $A' \subseteq A$, taki że:

$$\sum_{a_i \in A'} s(a_i) \leqslant b$$

gdzie maksymalizuje się łączną wartość elementów:

$$\sum_{a_i \in A'} w(a_i)$$

Złożoność obliczeniowa problemu plecakowego jest $O(n \cdot b)$, gdzie n jest liczbą elementów a b pojemnością plecaka.

2.7 Metaheurystyki

W przypadku problemów optymalizacyjnych trudnych obliczeniowo, znalezienie rozwiązania jest zwykle bardzo czasochłonne. W praktyce zdarza się, że wystarcza zaproponować rozwiązanie przybliżone, które może zostać znalezione w znacznie krótszym czasie.

Heurystyka jest metodą znajdowania rozwiązania problemu. Podejście to nie gwarantuje znalezienia rozwiązania optymalnego.

Metaheurystyki [36] to tzw. heurystyki nadrzędne, które sterują w procesie iteracyjnego przeszukiwania heurystykami niższego rzędu. Zaprojektowanie algorytmu metaheurystycznego dla badanego problemu wymaga doprecyzowania pewnych ogólnych pojęć związanych z daną metaheurystyką. Często metody te są inspirowane zjawiskami biologicznymi lub fizycznymi.

Przykładowe metaheurystyki to:

- Algorytm Ewolucyjny
- Metoda GRASP
- Metoda Tabu
- Algorytmy mrówkowe
- Symulowane Wyżarzanie

Algorytm ewolucyjny [4] to metaheurystyka, w której sposób przeszukiwania przestrzeni rozwiązań naśladuje procesy naturalne: dziedziczenie genetyczne i darwinowską metodę walki o przeżycie. Algorytm przetwarza populację osobników. Osobnik to pojedyncze rozwiązanie problemu. Każdy osobnik posiada genotyp, który określa fenotyp, podlegający ocenie środowiska. Środowisko opisane jest za pomocą funkcji przystosowania: każdemu osobnikowi przyporządkowuje się wartość liczbową określającą przystosowanie jego fenotypu do środowiska. Zaadaptowanie idei algorytmu ewolucyjnego do rozwiązania konkretnego problemu polega na zdefiniowaniu:

- sposobu kodowania

- środowiska (funkcji przystosowania)
- operatorów genetycznych
- schematu algorytmu

GRASP [41] to akronim Greedy Randomized Adaptive Search Procedure, jest metodą lokalnego przeszukiwania. Główną ideą jest stworzenie dobrego rozwiązania początkowego, a następnie jego lokalna optymalizacja. Do budowy rozwiązania początkowego używa się w GRASP specjalnej listy RLC (Restricted Candidate List - ograniczona lista kandydatów). RLC to lista elementów, które w danym momencie można wykorzystać do budowy rozwiązania. RLC posiada tylko najlepsze w danym momencie elementy. Na każdym etapie budowania rozwiązania lista RLC jest uaktualniana, w zależności od istniejącego cząstkowego rozwiązania początkowego. Następnie z listy RLC jest wybierany jeden element i dodawany do częściowego rozwiązania początkowego. Stosowane kryterium ograniczające liczbę kandydatów na liście RLC to przykładowo:

- wybór k najlepszych kandydatów;
- wybór wszystkich kandydatów spełniających postawione kryteria dotyczące jakości ich dopasowania do częściowego rozwiązania.

Zaadaptowanie tego podejścia do rozwiązania badanego problemu wymaga zdefiniowania:

- listy RLC
- sposobu kodowania
- metody lokalnego przeszukiwania

Rozdział 3

Podstawowe zagadnienia biologii molekularnej i chemii

3.1 Wprowadzenie

W rozdziale tym przedstawione zostaną podstawy biologii molekularnej i chemii niezbędne do zrozumienia istoty i znaczenia badanych problemów kombinatorycznych.

W rozprawie omówiono kompleksowe podejście do ustalania sekwencji peptydów: sekwencjonowanie i asemblację. Ten rozdział wprowadza czytelnika w zagadnienia dotyczące peptydów. Zostaje nakreślona ich budowa oraz te właściwości chemiczne, które zostaną wykorzystane w eksperymencie chemicznym oraz przy definiowaniu problemów kombinatorycznych.

Analizowany w niniejszej pracy problem kombinatoryczny budowy bibliotek oligonukleotydów związany jest z komplementarnością łańcuchów DNA. W rozdziale krótko przedstawiono budowę cząsteczek DNA oraz ich najważniejsze własności chemiczne, które są niezbędne do zrozumienia motywacji oraz zaproponowanych metod. Warto tutaj zaznaczyć, że rozważany w pracy problem związany z DNA (rozdział 4.) nie dotyczy bezpośrednio biologii molekularnej - nie jest związany z ustalaniem budowy cząsteczek, ich funkcji ani analizą i projektowaniem systemów biologicznych. Przedstawione metody mogą zostać wykorzystane w celu optymalizacji algorytmów na klasyczne komputery DNA.

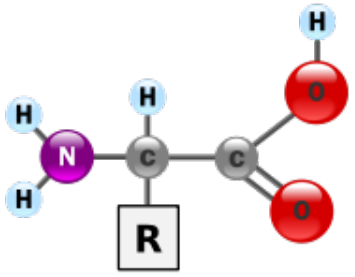
W rozdziale opisano również centralny dogmat biologii molekularnej, który pozwoli wyjaśnić znaczenie biologiczne badań dotyczących ustalenia budowy sekwencji peptydów.

Ze względu na charakter tej pracy, która jest rozprawą doktorską z nauk technicznych, wprowadzenie biologiczne napisane jest możliwie przystępnym językiem. Gdy jest to niezbędne, wprowadza się definicje zagadnień z dziedziny (np. jonizacja, pH). Głównym zamierzeniem autora jest, aby podane tutaj informacje były w pełni zrozumiałe i wyczerpujące dla osób, które nie posiadają wykształcenia chemicznego. Biologiczne wprowadzenie do komputerów DNA, sekwencjonowania oraz asemblacji zostało umieszczone odpowiednio w rozdziałach 4-6.

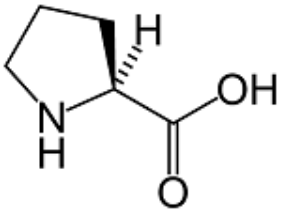
3.2 Peptydy, białka i aminokwasy

Białka składają się z 20 typów aminokwasów. **Aminokwasy** to związki chemiczne, które zawierają grupę aminową (-NH_2) oraz grupę karboksylową (-COOH)[23]. Ogólny wzór aminokwasu przedstawia rysunek 3.1. W zależności od budowy łańcucha bocznego R,

można wyróżnić 19 aminokwasów. Dodatkowy dwudziesty aminokwas, prolina, zawiera pięcioczłonowy pierścień i został przedstawiony na rysunku 3.2.



Rysunek 3.1: Ogólny wzór aminokwasu (źródło: Wikipedia)



Rysunek 3.2: Prolina (źródło: Wikipedia)

Wszystkie aminokwasy zestawiono w tabeli. Bardzo często korzysta się z trzyliterowego albo jednoliterowego skrótu nazw. Na potrzeby tej pracy proponuje się wykorzystanie jednoliterowego skrótu FASTA [załącznik A].

Aminokwasy można postrzegać jako cząsteczki z czterema podstawnikami przy węglu α. Trzy z nich są takie same dla wszystkich aminokwasów: gruba karboksylowa, grupa aminowa i atom węgla. Własności fizykochemiczne aminokwasów, szczególnie własności kwasowo-zasadowe, zależą od łańcucha bocznego R.

Na potrzeby dalszych definicji wprowadzone zostanie pojęcie substratu i produktu.

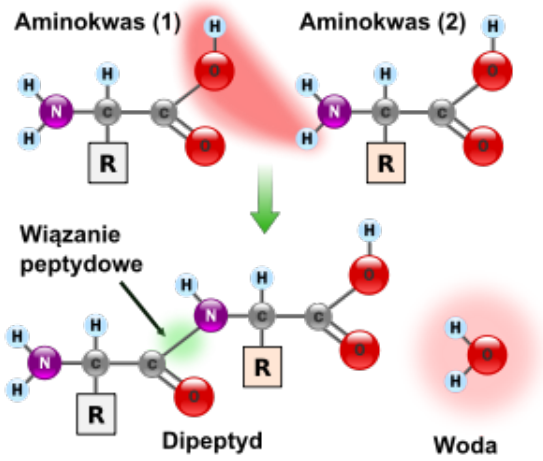
Substrat to związek chemiczny, który ulega przemianie w reakcji chemicznej, której wynikiem jest powstanie **produktu**.

Reakcja **kondensacji** to połączenie substratów ze sobą, którego wynikiem jest utworzenie większej cząsteczki produktu głównego, z dodatkowym wydzieleniem produktu ubocznego.

Dwa aminokwasy łączą się ze sobą za w reakcji kondensacji. Połączeniu ulega grupa karboksylowa jednego aminokwasu z grupą aminową drugiego. Wiązanie, które powstaje nazywane jest **wiązaniem peptydowym**. Wynikiem reakcji jest powstanie **dipeptydu** oraz wydzielenie się cząsteczki wody. Schemat tej reakcji przedstawiono na rysunku3.3.

Reakcja ta może zachodzić wielokrotnie, zawsze według powyższego schematu, gdzie uwspólnieniu ulega wolna grupa karboksylowa i wolna grupa aminowa. Związek powstały z połączenia w ten sposób wielu aminokwasów nazywany jest **peptydem**. W ten sposób powstają łańcuchy, które nie składają się jednak z całych aminokwasów, gdyż w reakcji wydziela się woda. Fragmenty aminokwasów wchodzące w skład peptydów noszą nazwę **reszt aminokwasowych**. Wśród peptydów wyróżnia się:

- Oligopeptydy, które zawierają kilka do kilkunastu reszt aminokwasowych



Rysunek 3.3: Dipeptyd - rysunek poglądowy (źródło: Wikipedia)

- Polipeptydy zawierające kilkadziesiąt reszt aminokwasowych
- Białka, są to bardzo długie cząsteczki

Peptydy są **polimerami** [1], czyli cząsteczkami w których wielokrotnie powtarzają się te same jednostki zwane merami (w przypadku peptydów są to aminokwasy).

Różnica między polipeptydem a białkiem nie jest jednoznacznie sprecyzowana. Rozróżnienie bazuje na masie cząsteczki rozważanego związku. Peptydy zawierające więcej niż 100 reszt aminokwasowych noszą nazwę **białek**[45].

Aminokwasy łącząc się ze sobą tworzą tzw, łańcuch główny. Na łańcuch główny składają się węgle α, przy których znajdują się pozostałe podstawniki. Ważną cechą peptydów jest to, że aminokwasy prawie zawsze łączą się w jeden główny łańcuch, nie tworząc rozgałęzień. Nawet jeśli łańcuchy boczne peptydów zawierają grupy karboksylowe, to bardzo rzadko zachodzi reakcja kondensacji z ich udziałem. Znane są peptydy, które posiadają rozgałęzienia, na przykład glutation. Peptyd ten nie powstaje jednak w typowej dla większości tych związków biosyntezie [23].

W cząsteczkach peptydów można wyróżnić dwa aminokwasy. Aminokwas, który posiada wolną grupę aminową, nazywany jest **N-końcowym** oraz aminokwas zawierający wolną grupę karboksylową, nazywany **C-końcowym**.

Powiedziano, że peptydy to połączone jedna za drugą reszty aminokwasowe w długi łańcuch. Kolejność aminokwasów w cząsteczce (sekwencja), ustalana począwszy od aminokwasu N-końcowego, nazywana jest **strukturą pierwszorzędową**.

W tej pracy skupiono się na badaniu struktury pierwszorzędowej peptydów, stąd używane w kolejnych rozdziałach pojęcia: struktura pierwszorzędowa, łańcuch aminokwasowy, długi peptyd, sekwencja aminokwasów, cząsteczka peptydu są równoznaczne.

Warto jeszcze dodać, że poznanie sekwencji aminokwasów, jakkolwiek bardzo ważne, jest wstępem do dalszych badań dotyczących peptydów. Ze względu na skalę przestrzenną, można wyróżnić cztery poziomy opisu długich peptydów (białek) [2]:

- Struktura pierwszorzędowa
- Struktura drugorzędowa - przestrzenne ułożenie łańcuchów
- Struktura trzeciorzędowa - wzajemne położenie elementów struktury drugorzędowej

- Struktura czwartorzędowa - wzajemne położenie łańcuchów i ewentualnych struktur niebiałkowych

W rozdziale 6. zaproponowano eksperyment chemiczny, którego częścią jest rozdzielenie mieszaniny krótkich peptydów. Poniżej opisano własność punktu izoelektrycznego peptydów i jego wykorzystania w elektroforezie, która pozwala rozdzielić taką mieszaninę. Wymaga to wprowadzenia pomocniczych definicji dotyczących jonizacji peptydów i pH.

Jonizacja to zjawisko powstania jonu z obojętnej cząsteczki związku. Jonizacja cząsteczki peptydu może zachodzić poprzez dołączenie się protonu do grupy aminowej (powstaje grupa NH_3^+) lub poprzez odłączenie się atomu węgla z grupy karboksylowej (powstaje grupa COO^-).

pH to ilościowa skala kwasowości i zasadowości roztworów wodnych związków. Przyjmuje wartości od 0 (najbardziej kwasowy) do 14 (najbardziej zasadowy). Wartość 7 oznacza roztwór obojętny.

W sekwencjach aminokwasowych większość grup NH_2 oraz COOH bierze udział w tworzeniu wiązań peptydowych. Część aminokwasów posiada jednak zasadowy charakter łańcuchów bocznych (jak lizyna czy arginina). Część posiada dodatkowe gruby karboksylowe (np. kwas asparaginowy). Badając peptydy w roztworze wodnym należy wziąć pod uwagę pH roztworu, które wpływa na ładunek, jakim zostanie obdarzona cząsteczka. W roztworach o niskim pH dodatkowe grupy aminowe mogą ulec protonowaniu do formy NH_3^+, natomiast w roztworach o wysokim pH grupy karboksylowe mogą ulec jonizacji do CO_2^-. Dla każdego peptydu istnieje taka wartość pH, przy której ładunek cząsteczki jest równy 0. Wartość ta nazywana jest **punktem izoelektrycznym (PI)**. Na wartość tą wpływa nie tylko liczba grup o charakterze kwasowym i zasadowym, ale również ich położenie w sekwencji [23].

Jeśli pH roztworu jest poniżej PI, to peptyd ma ładunek dodatni, jeśli powyżej, to peptyd ma ładunek ujemny. Zostało to wykorzystane w procesie elektroferezy.

Elektrofereza opiera się na obserwacji, iż przebyta droga cząsteczki peptydu w polu elektrycznym zależy od PI oraz masy tej cząsteczki. Dzięki temu mieszanina wielu krótkich peptydów może zostać rozdzielona na jednakowe frakcje [40].

3.3 Kwasy nukleinowe

Kwasy nukleinowe to polimery zbudowane z nukleotydów. Są to cząsteczki liniowe i pozbawione rozgałęzień. Można wyróżnić 2 typy kwasów nukleinowych:

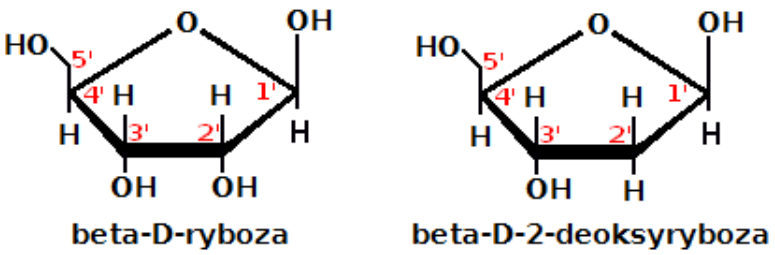
- Deoksyrybonukleinowe (DNA)
- Rybonukleinowe (RNA)

Nukleotydy zbudowane są z reszty cukrowej 3.4, zasady azotowej 3.5 oraz reszty fosforanowej. Przykładowy nukleotyd przedstawiono na 3.6.

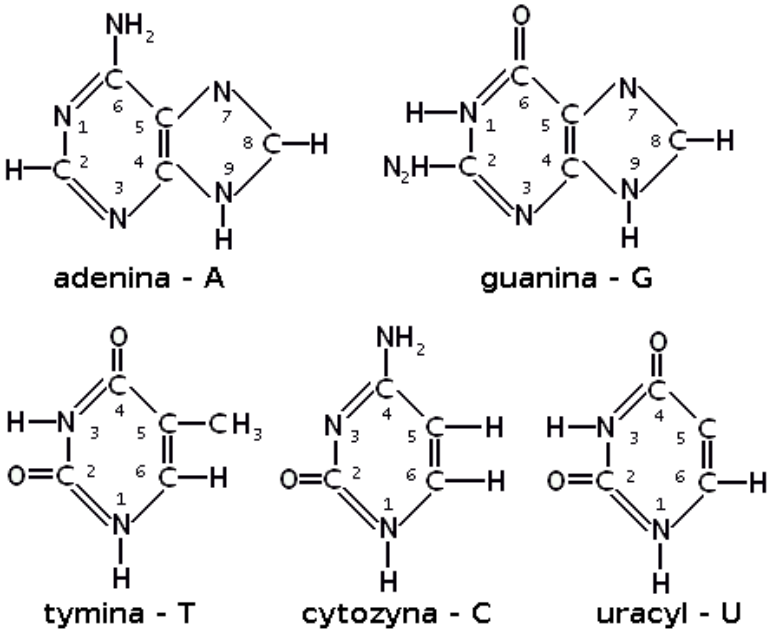
Cukier to ryboza w przypadku RNA albo deoksyryboza (w DNA).

W DNA występują 4 różne **zasady azotowe**:

- adenina (A)
- cytozyna (C)
- guanina (G)
- tymina (T)



Rysunek 3.4: Cukry (źródło: Wikipedia)



Rysunek 3.5: Zasady azotowe nukleotydów (źródło: Wikipedia)

W RNA tymina zostaje zastąpiona przez uracyl. Nukleotydy są rozpoznawane na podstawie zasady azotowej, którą zawierają.

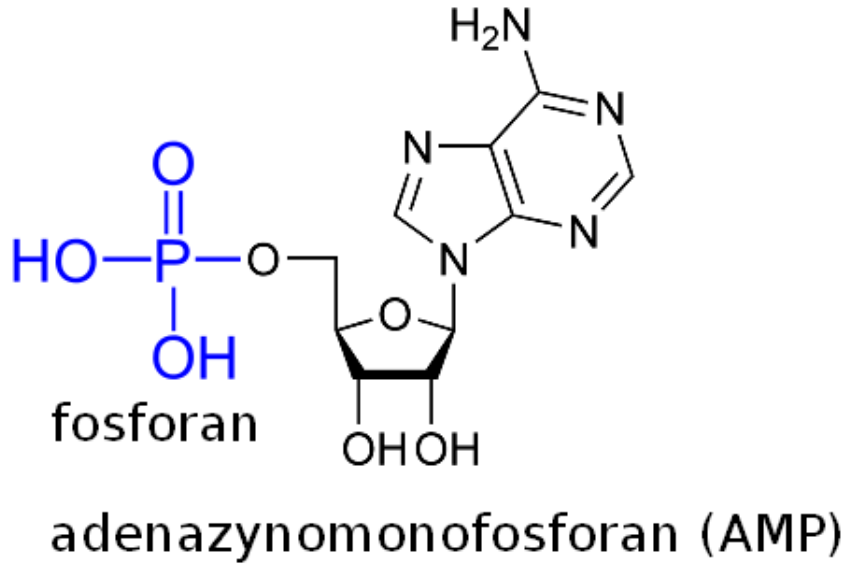
Nukleotydy mają zdolność łączenia się ze sobą wiązaniem fosfodiestrowym. W ten sposób tworzą się dłuższe cząsteczki. łańcuchy zawierające od kilkunastu do kilkudziesięciu nukleotydów noszą nazwę **oligonukleotydów**.

Łańcuchy DNA nie są symetryczne. Każda z nici ma z jednej strony wolną grupę fosforanową (tzw. **koniec 5'**) a z drugiej strony wolną grupę hydroksylową deoksyrybozy (tzw. **koniec 3'**).

Zasady azotowe wchodzące w skład nukleotydów mają zdolność łączenia się ze sobą za pomocą wiązania wodorowego, przy czym cytozyna może się połączyć jedynie z guainą (C-G), natomiast adenina z tyminą albo uracylem w przypadku RNA (A-T, A-U). Reguła ta nosi nazwę **zasady komplementarności**.

Cząsteczki DNA występują zwykle w postaci dwóch nici położonych antyrównolegle względem siebie - koniec 3' jednej jest położony nad końcem 5'. Odpowiadające sobie w łańcuchach kolejne nukleotydy są komplementarne i połączone wiązaniem wodorowym. Łańcuchy owijają się wokół wspólnej osi tworząc tak zwaną **podwójną helisę**.

Poniżej podano przykład dwóch komplementarnych nici DNA:



Rysunek 3.6: Nukleotyd A (źródło: Wikipedia)

```
5-CCTAGATGTGA-3
3-GGATCTACACT-5
```

3.3.1 Znaczenie biologiczne podjętych badań

Geny każdego organizmu żywego zawierają informację jak „zbudować" ten organizm. Cały materiał genetyczny organizmu nosi nazwę **genomu**. Jednym z największych osiągnięć ostatnich kilkunastu lat jest ustalenie budowy genomu człowieka.

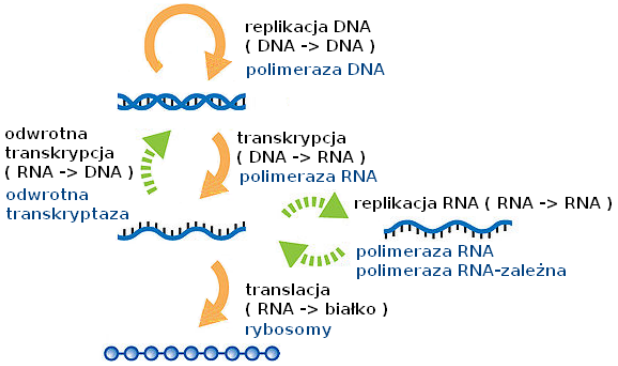
Według **centralnego dogmatu biologii molekularnej** [19], informacja zawarta w genach (DNA) przyjmuje ostatecznie fizyczną postać fenotypu (białka). Informacja zapisana w DNA w procesie transkrypcji jest przepisywana na jednoniciowe RNA. Następnie w procesie translacji, z udziałem rybosomów, z RNA powstaje białko 3.7. Każde 3 kolejne nukleotydy w łańcuchu DNA jednoznacznie określają 1 aminokwas w białku. Wynika z tego, że poznanie genomu ludzkiego jest kluczowe, aby określić ekspresję białek w organizmie. Cechy organizmów, takie jak ich budowa, fizjologia czy nawet zachowanie (instynkty zwierzęce, ludzkie talenty i skłonności) są wynikiem występowania w komórkach odpowiednich białek.

Poniżej przedstawiono wybrane funkcje białek w organizmie [23]:

- katalizują reakcje (enzymy)
- transportują inne cząsteczki w organizmie (np. hemoglobina transportująca tlen)
- regulują procesy transkrypcji i ekspresji poszczególnych genów
- wykonują pracę związaną ze skurczem mięśni

Ważna dla omówionego procesu jest **replikacja DNA** w trakcie której, przy udziale enzymu zwanego polimerazą DNA, następuje powielenie łańcuchów.

Warto również wspomnieć, że w wirusach zaobserwowano dwa inne przejścia: odwrotna transkrypcja (RNA w DNA) oraz replikacja RNA. Dodatkowo w warunkach laboratoryjnych udało się wykonać bezpośrednią translację DNA do białka [48].



RYSUNEK 3.7: Centralny dogmat biologii molekularnej (źródło: Wikipedia)

Określenie budowy genomu pozwala na określenie białek, które powstają w procesie transkrypcji, a następnie translacji. W praktyce jednak, po procesie translacji, w białkach może wystąpić wiele **modyfikacji potranslacyjnych**, które mają wpływ na budowę białek, a co za tym idzie na ich funkcje. Oznacza to, że analiza genomu nie jest wystarczająca, aby określić fenotyp (białko). Jak pokazuje tabela (Załącznik A.2) znanych jest ponad 150 modyfikacji białek. Istnieje zatem potrzeba skonstruowania metod, które pozwolą na określenie sekwencji białek.

Rozdział 4

Biblioteki oligonukleotydów antykomplementarnych

4.1 Wprowadzenie i motywacja

Biblioteki DNA są zbiorami łańcuchów DNA o pewnych własnościach. Zwykle reprezentują one materiał genetyczny pochodzący z żywych organizmów i otrzymywane są z eksperymentów biochemicznych [50]. Stosowane są np. do określania ekspresji genów (w przypadku bibliotek cDNA, składających się z łańcuchów DNA po odwrotnej transkrypcji z mRNA).

Rozważane w niniejszym rozdziale biblioteki mają odmienne własności i zastosowanie, które wywodzi się z koncepcji komputerów DNA. Model obliczeń nazwany „DNA computing", który oparty jest o reakcje chemiczne, został zaprezentowany w 1994 roku przez Adlemana [3]. Zaproponowane zostało rozwiązanie problemu ścieżki Hamiltona w grafie przy wykorzystaniu podstawowych operacji biochemicznych oraz metod laboratoryjnych takich jak:

- hybrydyzacja DNA,
- ligacja - łączenie łańcuchów DNA między końcem 3' a końcem 5' ,
- łańcuchowa reakcja polimeryzacji (PCR) - powielanie cząsteczek DNA,
- system kulek magnetycznych - wyodrębnianie łańcuchów z roztworu, które zawierają daną sekwencję,
- elektroforeza na żelu - określenie długości łańcuchów DNA.

Reprezentacja grafu $G = (V, A)$ opiera się na losowo generowanych łańcuchach DNA o parzystej długości l dla każdego wierzchołka z V. Łuki są reprezentowane również jako łańcuchy DNA o długości l, jednak ich sekwencja jest zdeterminowana przez możliwość hybrydyzacji do odpowiednich wierzchołków. Jeśli $x, y \in V$ oraz $(x, y) \in A$, to łuk jest fizycznie tworzony przez hybrydyzację drugiej połowy nukleotydów łańcucha reprezentującego wierzchołek x z pierwszą połową nukleotydów łańcucha reprezentującego łuk (x, y). Podobnie, jego druga połowa hybrydyzuje z pierwszą połową łańcucha reprezentującego wierzchołek y. Poniższy przykład ilustruje ten proces:

```
x = ACTTGTGCCG
y = GTGCGAACGT
(x,y) = ACGGCCACGC
```

```
    ACGGCCACGC
ACTTGTGCCGGTGCGAACGT
```

Łańcuchy reprezentujące łuki oraz wierzchołki są nierozróżnialne pod względem biochemicznym, co powoduje że możliwa jest niechciana hybrydyzacja między wierzchołkami lub między łukami. Taka hybrydyzacja powoduje utratę materiału DNA przez co obniża prawdopodobieństwo znalezienia prawidłowego rozwiązania. Ograniczenie takiej hybrydyzacji może nastąpić więc poprzez wykorzystanie do kodowania wierzchołków zbioru łańcuchów, które mają zminimalizowane prawdopodobieństwo hybrydyzacji między sobą. Losowy sposób generowania tych łańcuchów nie minimalizuje tego prawdopodobieństwa, gdyż dopuszczalne jest wylosowanie łańcuchów, które są w całości komplementarne. Aby ograniczyć prawdopodobieństwo hybrydyzacji w zbiorze łańcuchów DNA, zaproponowany został algorytm budowy biblioteki oligonukleotydów (opisany także w [34]), które mają możliwie niewielką tendencję do hybrydyzacji między sobą. Wykorzystanie algorytmu optymalizuje więc sposób kodowania instancji problemu ścieżki Hamiltona dla komputerów DNA.

4.2 Definicja problemu i metoda rozwiązania

4.2.1 Reprezentacja

Zgodnie z powyższymi rozważaniami, optymalizacja sposobu kodowania instancji dla problemu ścieżki Hamiltona wymaga utworzona biblioteki oligonukleotydów, dla której prawdopodobieństwo hybrydyzacji między dowolną parą oligonukleotydów jest minimalizowane.

DNA może być reprezentowane jako czteroliterowy alfabet, ale sam algorytm można uogólnić na dowolny alfabet o parzystej liczbie symboli. Wymagane jest, żeby alfabet miał parzystą liczbę symboli, gdyż tylko wtedy można zdefiniować analogiczną do DNA **binarną relację komplementarności**. Z uwagi na generalizację mocy alfabetu elementy biblioteki będą zwane dalej łańcuchami.

Wszystkie wierzchołki w eksperymencie są reprezentowane przez łańcuchy DNA o tej samej długości. Parametry takiej biblioteki można więc ograniczyć do dwóch zmiennych:

- l - długość pojedynczego łańcucha
- n - liczba łańcuchów

Zaproponowany algorytm opiera się na założeniu, że uzyskanie równego prawdopodobieństwa hybrydyzacji między dowolną parą elementów biblioteki może być otrzymane poprzez budowę jednego łańcucha o długości przynajmniej równej nl, którego sekwencja nukleotydów jest skonstruowana w taki sposób, że jego dowolne podłańcuchy o ustalonej długości mają te same zminimalizowane prawdopodobieństwa hybrydyzacji. Łańcuch o takich własnościach może być następnie pocięty na n części o długości l tworzących wynikową bibliotekę.

Należy ponadto zdefiniować kryterium wg. którego dwa łańcuchy kwalifikowane są jako niekomplementarne. Przyjęte zostało naturalne kryterium, że dwa łańcuchy o tej samej długości są niekomplementarne, jeśli nie są w pełni komplementarne tj. przynajmniej na jednej pozycji nie zachodzi hybrydyzacja – mówimy wtedy, że takie łańcuchy są **antykomplementarne**. Można zauważyć, że dla łańcucha DNA o długości k istnieje $4^k - 1$ łańcuchów antykomplementarnych (gdyż tylko jeden jest komplementarny spośród

4^k możliwych sekwencji). Zaproponowany algorytm budowy biblioteki daje pewną swobodę w wyborze pomiędzy łańcuchami antykomplementarnymi, więc spełnienie warunku antykomplementarności może być punktem wyjściowym dla metody wyboru konkretnego łańcucha (opartej np. o termodynamiczny model siły wiązań).

Można zauważyć, że konstrukcja biblioteki przy użyciu wyłącznie liter, które nie są komplementarne prowadzi do całkowitego wyeliminowania hybrydyzacji między jej elementami. Jest to jednak przypadek trywialny, który nie wymaga stosowania skomplikowanego algorytmu, a ponadto ma następujące wady z punktu widzenia zastosowania do kodowania instancji w komputerze DNA:

- powoduje konieczność wydłużenia łańcucha dla konstrukcji tej samej liczby łańcuchów z uwagi na użycie o połowę mniejszego alfabetu,
- powoduje zwiększenie prawdopodobieństwa błędnych hybrydyzacji łuków, gdyż z uwagi na mniejszy alfabet podłańcuchy o tej samej długości będą się powtarzać częściej między różnymi elementami biblioteki (np. wierzchołki ACCCCC i CCCCCA mają wspólny podłańcuch CCCCC – przykładowy łuk GGGGGT może hybrydyzować z wierzchołkiem CCCCCA w sposób pełny i prawidłowy lub sposób częściowy i błędny z wierzchołkiem ACCCCC).

Powyższe wady mają szczególne znaczenie dla alfabetu czteroliterowego, na którym bazuje DNA, gdyż w tym przypadku zmniejszenie mocy alfabetu o połowę spowoduje nawet dwukrotny wzrost długości łańcucha przy zachowaniu takiej samej liczby elementów biblioteki – wynika to z faktu, że $\lceil \log_a b \rceil$ jest długością, która jest potrzebna do zakodowania b różnych łańcuchów nad alfabetem o mocy a, więc dla DNA i liczby łańcuchów n widać, że zmniejszenie alfabetu o połowę powoduje dwukrotne wydłużenie łańcucha:

$$\lceil \log_4 n \rceil = \left\lceil \frac{\log_2 n}{\log_2 4} \right\rceil = \left\lceil \frac{\log_2 n}{2} \right\rceil$$

Z uwagi na powyższe fakty, zakłada się, że elementy biblioteki mogą się składać z liter całego alfabetu i każda litera ma równe prawdopodobieństwo wystąpienia.

4.2.2 Definicja problemu

W celu formalnego zdefiniowania problemu konstrukcji biblioteki oligonukleotydów antykomplementarnych wprowadzone zostaną następujące oznaczenia:
Notacja. Niech Σ będzie alfabetem oraz $a, b \in \Sigma^*$ będą dowolnymi słowami nad tym alfabetem. Wtedy:

- $|a|$ jest długością słowa a,
- $l_k(a)$ jest k-tą literą słowa a, tym samym $a = \Big(l_1(a), \ldots, l_{|a|}(a)\Big)$,
- $a \sqsubset b$ oznacza, że słowo b zawiera słowo a.

Podsumowując powyższe rozważania, problem konstrukcji biblioteki oligonukleotydów antykomplementarnych można zdefiniować formalnie w następujący sposób:
INSTANCJA:

- liczba elementów biblioteki n oraz długość każdego elementu l,
- alfabet Σ, $\quad |\Sigma| = \alpha, \quad 2|\alpha$,
- relacja komplementarności $\sim$ w zbiorze niepustych słów nad alfabetem Σ, tj. w zbiorze $U = \Sigma^* \backslash \{\lambda\}$, o następujących własnościach:

1) dla liter alfabetu Σ:
   a) jest symetryczna, tj. $\forall_{a,b\in\Sigma}\ a \sim b \Rightarrow b \sim a$
   b) jest przeciwzwrotna, tj. $\forall_{a,b\in\Sigma}\ a \sim b \Rightarrow a \neq b$
   c) każdy element ze zbioru Σ ma dokładnie jeden element komplementarny tj.
   $\forall_{a\in\Sigma}\ |\{b : b \in \Sigma \wedge a \sim b\}| = 1$
2) słowa (przynajmniej dwuliterowe, tj. ze zbioru $U\backslash\Sigma$) są komplementarne, jeśli są tej samej długości oraz litery na odpowiednich przeciwległych pozycjach są komplementarne tj.

$$\forall_{a=(l_1(a),\ldots,l_{|a|}(a)),b=(l_1(b),\ldots,l_{|b|}(b))\in U\backslash\Sigma}$$
$$a \sim b \Leftrightarrow |a| = |b| \wedge \forall_{i\in\{1,\ldots,|a|\}}\, l_i(a) \sim l_{|a|-i+1}(b)$$

   (można zauważyć, że sposób zdefiniowania relacji komplementarności słów powoduje zachowanie własności 1a-1c dla pojedynczych liter)
3) jeśli słowa o tej samej długości nie są komplementarne, to nazywamy je antykomplementarnymi i mówimy, że zachodzi między nimi relacja $\not\sim$:

$$\forall_{a=(l_1(a),\ldots,l_{|a|}(a)),\ b=(l_1(b),\ldots,l_{|b|}(b))\in U}$$
$$a \not\sim b \Leftrightarrow |a| = |b| \wedge \neg\,(a \sim b)$$

ODPOWIEDŹ: Słowo P o długości $|P| \geqslant p = nl$ nad alfabetem Σ takie, że dowolne jego podsłowa o długości $j = f(p)$ (gdzie f jest funkcją wyznaczającą minimalną wartość j w zależności od p) będą antykomplementarne:

$$\forall_{a,b\sqsubset P\ \wedge\ |a|=|b|=j}\ a \not\sim b \tag{4.1}$$

Ponadto, wymagane jest, aby każda litera alfabetu Σ miała równe prawdopodobieństwo występowania w P.

4.2.3 Algorytm

Dla problemu opisanego powyżej zaproponowano następujący algorytm:

1. Utwórz graf de Bruijna $B_k^\alpha = (V, A)$ (posiada α^k wierzchołków oraz α^{k+1} łuków), gdzie $k = j - 1 = f(p) - 1$ oraz $p = nl$ (postać funkcji f będzie zdefiniowana dalej) oraz:

   - dla każdego wierzchołka $x \in V$ niech $[x]$ oznacza jego etykietę, która jest słowem ze zbioru Σ^k zgodnie z definicją grafu de Bruijna,
   - dla każdego łuku $a = (x, y) \in A$ niech $[a]$ oznacza jego etykietę, która jest połączeniem etykiety poprzednika i następnika z jednokrotnym uwzględnieniem nakładającej się części tj.

$$a = (x, y) \in A \Rightarrow [a] = ([x], l_k(y)) = (l_1(x), [y]) \Rightarrow [a] \in \Sigma^j$$

2. Dla każdej komplementarnej pary łuków $a_1, a_2 \in A$, tj. takiej, że:

$$[a_1] \sim [a_2]$$

   usuń a_1 lub a_2. Metoda usuwania (opisana w dalszej części) musi zapewniać, że otrzymany graf $C = (V, A')$ będzie grafem Eulera.

3. Znajdź obwód Eulera $E = (a_1, a_2, a_3, \ldots, a_m)$ w grafie $C = (V, A')$ (gdzie $\forall_{i \in \{1,\ldots,m\}}\, a_i \in A'$ i $|A'| = m = \frac{|A|}{2} = \frac{\alpha^j}{2}$).

4. Zbuduj wynikowy łańcuch P przy użyciu znalezionego obwodu Eulera $E = (a_1, a_2, a_3, \ldots, a_m)$ z kolejno dołączanych ostatnich liter etykiet łuków tj.:

$$P = (l_j(a_1), l_j(a_2), \ldots, l_j(a_{m-1}), l_j(a_m))$$

5. Podziel łańcuch P na n części o długości l. Z uwagi na fakt, że $|P| \geqslant p = nl$ (wyjaśnienie przy definicji funkcji $f(p)$) należy pominąć ewentualny nadmiarowy fragment łańcucha.

Algorytm gwarantuje, że każda para s_1, s_2 podłańcuchów P o długości $|s_1| = |s_2| = j$ będzie spełniała zależność:

$$s_1 \not\sim s_2$$

Wynika to z faktu, że każdy taki podłańcuch jest obrazem pewnego łuku, dla którego został usunięty łuk komplementarny.

Należy wyjaśnić jeszcze dwa zagadnienia:

a) obliczenie parametru j,

b) metodę usuwania łuków komplementarnych gwarantującą zachowanie obwodu Eulera.

4.2.4 Wyznaczanie parametrów grafu

Aby elementy biblioteki miały minimalną tendencję do hybrydyzacji, relacja $s_1 \not\sim s_2$ powinna się opierać na możliwie najkrótszych łańcuchach, czyli należy znaleźć minimalną wartość j (a zatem też k), dla której możliwe jest utworzenie wynikowego łańcucha o długości p.

Sposób budowy łańcucha wynikowego z obwodu Eulera powoduje, że łańcuch ma taką samą długość, co liczba łuków, czyli:

$$|P| = m = \frac{\alpha^j}{2} \;\Rightarrow\; \alpha^j = 2\,|P| \;\Rightarrow\; j = \log_\alpha 2\,|P|$$

Łańcuch P ma mieć długość jak najbliższą parametrowi $p = nl$, należy więc podstawić p w miejsce $|P|$ i zaokrąglić wynik logarytmu w górę, gdyż wartość $2p$ nie musi być potęgą α, a długość etykiety musi być liczbą całkowitą:

$$j = \log_\alpha 2\,|P| = \lceil \log_\alpha 2p \rceil = \lceil \log_\alpha 2nl \rceil \tag{4.2}$$

Zatem $k = j - 1 = \lceil \log_\alpha 2nl \rceil - 1$, przy czym zaokrąglenie w górę powoduje, że wynikowy łańcuch P może mieć nadmiarową długość.

4.2.5 Metoda usuwania łuków komplementarnych

W tym podrozdziale omówiona jest metoda usuwania łuków z grafu de Bruijna $B(\alpha, k) = (V, A)$ w taki sposób, że otrzymany graf $C = (V, A')$ jest grafem Eulera, a zbiór A' nie zawiera łuków komplementarnych. W celu wyjaśnienia tej metody zostanie wprowadzona następująca notacja:

$\forall_{a=(v_1,v_2)\in A}$:

- baza łuku a: $base(a) = (l_2(v_1), \ldots, l_k(v_1)) = (l_1(v_2), \ldots, l_{k-1}(v_2))$

- swoboda lewostronna łuku a: $left(a) = l_1(v_1)$

- swoboda prawostronna łuku a: $right(a) = l_k(v_2)$

Można zauważyć, że zbiór wszystkich możliwych wartości baz łuków w grafie $B(\alpha, k)$, jest równy Σ^{k-1} (gdzie Σ jest alfabetem o mocy α). Każdy element tego zbioru zawiera więc swój element komplementarny. Pozwala to na podział zbioru Σ^{k-1} na dwa podzbiory Q i W, wewnątrz których żadne dwa elementy nie są komplementarne tj.

$$\{base\,(a) : a \in A\} = \Sigma^{k-1} = Q \cup W$$
$$\text{oraz } \forall_{b_1,b_2 \in Q}\, b_1 \not\sim b_2 \;\; \wedge \;\; \forall_{b_1,b_2 \in W}\, b_1 \not\sim b_2$$

zbiory Q i W mają więc następujące własności:

$$|Q| = |W| = \frac{\alpha^{k-1}}{2} \qquad\qquad Q \cap W = \emptyset$$

Można zauważyć także, że graf $B(\alpha, k) = (V, A)$ ma następujące własności:

1) każdy wierzchołek ma taki sam stopień wejściowy i wyjściowy, równy mocy alfabetu tj. $\forall_{v \in V} out(v) = in(v) = \alpha$,

2) dla każdej bazy łuku zbiór utworzony ze swobód lewo/prawostronnych łuków opartych na tej bazie jest identyczny i równy alfabetowi, tj.

$$\forall_{b \in \Sigma^{k-1}} \{left\,(a) : a \in A \;\wedge\; base\,(a) = b\} = \{right\,(a) : a \in A \;\wedge\; base\,(a) = b\} = \Sigma$$

Powyższe własności pokazują bardzo silną symetrię grafu, która pozwoli na globalne podejście do sposobu usuwania komplementarnych łuków w taki sposób, aby zachować obwód Eulera.

Dysponując zbiorami Q oraz W należy wybrać jeden z nich, np. Q i następnie dla niego dokonywać wyboru, czy łuki o danej bazie i wybranych swobodach lewo/prawostronnych zostają w grafie, czy nie – dokonanie wyboru dla zbioru Q implikuje dokonanie przeciwnego wyboru dla łuku komplementarnego opartego na bazie ze zbioru W.

Aby zachować własność obwodu Eulera, należy dla każdego wierzchołka zawsze usunąć taką samą liczbę łuków wchodzących co łuków wychodzących. Można to zagwarantować przy użyciu schematu decyzyjnego wspólnego dla wszystkich łuków opartych na wybranym zbiorze baz (Q lub W), który oparty byłby na wyżej opisanej własności identyczności zbioru swobód dla każdej bazy łuku. Swobody lewostronne to elementy, które odróżniają etykiety poprzedników w łukach opartych na danej bazie, natomiast swobody prawostronne to elementy, które odróżniają etykiety następników. Jeśli dla każdej bazy łuku zbiór swobód lewo- i prawostronnych jest identyczny, to fakt ten może być użyty do globalnego podejścia do podejmowania decyzji o usunięciu/pozostawieniu łuku w grafie. Wystarczy bowiem, że dla każdej bazy z wybranego zbioru (Q lub W) podejmiemy decyzję o usunięciu dokładnie połowy łuków wchodzących i połowy łuków wychodzących – zachowamy wtedy warunek istnienia obwodu Eulera. Jednocześnie, dokonanie wyboru dla bazy z wybranego zbioru powoduje dokonanie przeciwnego wyboru dla komplementarnej bazy z drugiego zbioru.

Schemat postępowania może więc być zapisany następująco:

1. Podziel zbiór wszystkich baz łuków na dwa podzbiory Q i W takie, że wewnątrz każdego z nich nie istnieją dwie bazy komplementarne.

2. Zbuduj tablicę decyzyjną t o rozmiarach $\alpha \times \alpha$ w której kolumny odpowiadają możliwym wartościom swobód prawostronnych, a wierszach odpowiadają możliwym wartościom swobód lewostronnych (każda komórka tablicy reprezentuje więc decyzję dotyczącą usunięcia łuku o danej swobodzie lewo- i prawostronnej). Wybierz w tej tablicy komórki w taki sposób, aby w każdym wierszu i w każdej kolumnie wybrana była dokładnie połowa komórek. Tablicę tą można zatem opisać jako funkcję o następujących własnościach:

$$t : \Sigma^2 \to \{0,1\} \qquad \forall_{r\in\Sigma} \sum_{i\in\Sigma} t\left(i,r\right) = \frac{\alpha}{2} \qquad \forall_{r\in\Sigma} \sum_{i\in\Sigma} t\left(r,i\right) = \frac{\alpha}{2}$$

Każda konkretna funkcja spełniająca powyższą własność jest poprawna, a sposób jej doboru dla DNA może być przedmiotem badań dotyczących minimalizacji siły wiązań opartych na termodynamicznych modelach DNA.

3. Wybierz zbiór, dla którego będą podejmowane decyzje na podstawie tablicy t. Bez szkody dla ogólności rozumowania można założyć, że to będzie zbiór Q.

4. Usuń z grafu $B(\alpha,k) = (V,A)$ łuki, które posiadają następujące własności:

$$\begin{aligned} & a \in A \wedge \\ & \quad ((base\left(a\right) \in Q \ \wedge \ t\left(left\left(a\right), right\left(a\right)\right) = 1) \quad \vee \\ & \quad (base\left(\tilde{a}\right) \in Q \ \wedge \ t\left(left\left(\tilde{a}\right), right\left(\tilde{a}\right)\right) = 0)) \end{aligned}$$

gdzie $\tilde{a}$ oznacza łuk komplementarny do a.

Dla wyjaśnienia działania algorytmu zostanie omówiony przykład:

Należy zbudować bibliotekę oligonukleotydów antykomplementarnych składającą się z $n = 6$ łańcuchów o długości $l = 5$ nukleotydów.

Z treści zadania można wyznaczyć parametry:

1) $p = nl = 30$,

2) alfabetem jest DNA, czyli:

- $\alpha = 4$,
- $\Sigma = \{A, C, T, G\}$
- relacja $\sim$ jest zdefiniowana jako $A \sim T$ oraz $C \sim G$

3) $j = \lceil \log_\alpha 2p \rceil = \lceil \log_4 60 \rceil = 3$

4) $k = j - 1 = 2$

Zgodnie z kolejnymi krokami algorytmu:

1. Budowany jest graf de Bruijna $B(4,2) = (V,A)$. Macierz sąsiedztwa tego grafu pokazana jest w tabeli 4.2.

2. Usuwane są łuki komplementarne z $B(4,2)$ z zachowaniem obwodu Eulera w celu uzyskania grafu $C = (V,A')$. W tym celu wykonywane są następujące kroki:

   a) wyznaczane są zbiory $Q, W \in \Sigma^{k-1}$, czyli np. $Q = \{T,C\}$, $W = \{A,G\}$,

Bazy łuków	Zbiory usuwanych łuków (zbiory podkreślone falą zawierają łuki komplementarne do usuwanych i są wypisane w celach pomocniczych)
$base(a) = T \in Q$:	$\{a \in A : base(a) = T \ \wedge \ t\,(left(a), right(a)) = 1\} \ = \{ATA, ATC, CTA, CTC, TTT, TTG, GTT, GTG\}$
$base(a) = C \in Q$:	$\{a \in A : base(a) = C \ \wedge \ t\,(left(a), right(a)) = 1\} \ = \{ACA, ACC, CCA, CCC, TCT, TCG, GCT, GCG\}$
$base(\tilde{a}) = T \in Q$: $\Downarrow$ $base(a) = A \in W$:	$\{\tilde{a} \in A : base(\tilde{a}) = T \ \wedge \ t\,(left(\tilde{a}), right(\tilde{a})) = 0\} \ = \{TTA, TTC, GTA, GTC, ATT, CTT, CTG, ATG\}$ $\Downarrow$ $\{a \in A : base(a) = A \ \wedge \ t\,(left(\tilde{a}), right(\tilde{a})) = 0\} \ = \{TAA, GAA, TAC, GAC, AAT, AAG, CAG, CAT\}$
$base(\tilde{a}) = C \in Q$: $\Downarrow$ $base(a) = G \in W$:	$\{\tilde{a} \in A : base(\tilde{a}) = C \ \wedge \ t\,(left(\tilde{a}), right(\tilde{a})) = 0\} \ = \{TCA, TCC, GCA, GCC, ACT, CCT, CCG, ACG\}$ $\Downarrow$ $\{a \in A : base(a) = G \ \wedge \ t\,(left(\tilde{a}), right(\tilde{a})) = 0\} \ = \{TGA, GGA, TGC, GGC, AGT, AGG, CGG, CGT\}$

TABELA 4.1: Decyzje o usunięciu łuków w zależności od bazy łuku oraz wartości w tabeli decyzyjnej.

b) wyznaczana jest dowolna tablica decyzyjna $t : \Sigma^2 \to \{0, 1\}$, w której suma wartości w dowolnym wierszu i dowolnej kolumnie będzie równa $\frac{\alpha}{2} = 2$, np.:

	A	C	T	G
A	1	1	0	0
C	1	1	0	0
T	0	0	1	1
G	0	0	1	1

c) usuwane są łuki z A zgodnie z formułą:

$$\begin{aligned} & a \in A \wedge \\ & \quad ((base\,(a) \in Q \ \wedge \ t\,(left\,(a)\,, right\,(a)) = 1) \quad \vee \\ & \quad (base\,(\tilde{a}) \in Q \ \wedge \ t\,(left\,(\tilde{a})\,, right\,(\tilde{a})) = 0)) \end{aligned}$$

Wynikiem tego działania jest graf $C = (V, A')$ o macierzy sąsiedztwa przedstawionej w tabeli 4.3. Tabela 4.1 przedstawia decyzje o usunięciu łuków podjęte zgodnie z powyższym schematem.

3. W grafie $C = (V, A')$ wyszukiwany jest obwód Eulera np.

$$\begin{aligned} & E = \\ & AA \to AA \to AC \to CT \to TT \to TC \to CA \to AC \to CG \to \\ & GC \to CC \to CT \to TG \to GT \to TC \to CC \to CG \to GA \to \\ & AT \to TT \to TA \to AT \to TG \to GG \to GG \to GT \to TA \to \\ & AG \to GA \to AG \to GC \to CA \to AA \end{aligned}$$

4. Na podstawie obwodu E budowany jest łańcuch wynikowy P zgodnie z formułą:

$$P = (l_j(a_1), l_j(a_2), \ldots, l_j(a_{m-1}), l_j(a_m))$$

czyli $P = ACTTCACGCCTGTCCGATTATGGGTAGAGCAA$

	AA	AC	AT	AG	CA	CC	CT	CG	TA	TC	TT	TG	GA	GC	GT	GG
AA	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
AC	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
AT	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
AG	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
CA	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
CC	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
CT	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
CG	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
TA	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
TC	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
TT	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
TG	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
GA	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
GC	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
GT	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
GG	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Tabela 4.2: Macierz sąsiedztwa dla grafu $B(4,2) = (V, A)$

	AA	AC	AT	AG	CA	CC	CT	CG	TA	TC	TT	TG	GA	GC	GT	GG
AA	1	1	@	@	0	0	0	0	0	0	0	0	0	0	0	0
AC	0	0	0	0	@	@	1	1	0	0	0	0	0	0	0	0
AT	0	0	0	0	0	0	0	0	@	@	1	1	0	0	0	0
AG	0	0	0	0	0	0	0	0	0	0	0	0	1	1	@	@
CA	1	1	@	@	0	0	0	0	0	0	0	0	0	0	0	0
CC	0	0	0	0	@	@	1	1	0	0	0	0	0	0	0	0
CT	0	0	0	0	0	0	0	0	@	@	1	1	0	0	0	0
CG	0	0	0	0	0	0	0	0	0	0	0	0	1	1	@	@
TA	@	@	1	1	0	0	0	0	0	0	0	0	0	0	0	0
TC	0	0	0	0	1	1	@	@	0	0	0	0	0	0	0	0
TT	0	0	0	0	0	0	0	0	1	1	@	@	0	0	0	0
TG	0	0	0	0	0	0	0	0	0	0	0	0	@	@	1	1
GA	@	@	1	1	0	0	0	0	0	0	0	0	0	0	0	0
GC	0	0	0	0	1	1	@	@	0	0	0	0	0	0	0	0
GT	0	0	0	0	0	0	0	0	1	1	@	@	0	0	0	0
GG	0	0	0	0	0	0	0	0	0	0	0	0	@	@	1	1

Tabela 4.3: Macierz sąsiedztwa dla grafu $C = (V, A')$. Symbolem @ oznaczone są usunięte łuki.

5. Łańcuch P jest cięty na $n = 6$ łańcuchów o długości $l = 5$ z pominięciem nadmiarowych dwóch nukleotydów. Otrzymana biblioteka ma zatem postać:

$$\{ACTTC, ACGCC, TGTCC, GATTA, TGGGT, AGAGC\}$$

Rozdział 5

Sekwencjonowanie łańcuchów peptydowych

5.1 Wstęp

Jak wspomniano wcześniej, peptydy to związki składające się z 20 typów aminokwasów połączonych wiązaniami peptydowymi w długie łańcuchy. Ustalanie kolejności aminokwasów w cząsteczce nosi nazwę sekwencjonowania. Dwie najbardziej popularne metody degradacja Edmana i spektrometria masowa.

Degradacja Edmana jest metodą analityczną, która pozwala poznać N-terminalny aminokwas w badanej sekwencji. Aby uzyskać informację o sekwencji aminokwasów, reakcję tę przeprowadza się cyklicznie. Wynikiem pojedynczej reakcji jest utworzenie związku zawierającego N-terminalny aminokwas. Związek ten poddawany jest dodatkowej analizie w celu określenia rodzaju aminokwasu (np. za pomocą elektroforezy). W praktyce metodę tę można wykorzystać do określenia jedynie krótkich sekwencji. Dodatkowo do analizy wymagana jest relatywnie duża ilość oczyszczonego peptydu.

W eksperymencie nie są generowane żadne dodatkowe informacje wymagające przetworzenia lub interpretacji. Degradacja Edmana nie jest więc inspiracją do powstania modelu matematycznego ani algorytmów. Mechanizm tej reakcji zostanie jednak omówiony w tym rozdziale, gdyż wyniki eksperymentu z wykorzystaniem degradacji Edmana, podobnie jak wyniki eksperymentu przeprowadzonego za pomocą spektrometru, mogą stanowić dane wejściowe dla problemu asemblacji omówionego w kolejnym rozdziale.

Spektrometria masowa to technika polegająca na jonizacji i fragmentacji cząsteczki w spektrometrze masowym, a następnie na rozdzieleniu jonów fragmentarycznych według stosunku ich masy do ładunku (m/z) i analizie tak powstałej informacji. Wyniki ze spektrometru są przedstawiane w postaci tzw. widma masowego.

Zasadniczą zaletą tej metody w stosunku do degradacji Edmana jest mniejsze wymaganie odnośnie wielkości badanej próbki. Widmo masowe jest jednak informacją stosunkowo złożoną i wymaga dodatkowej interpretacji. Można je wykorzystać do rozpoznania znanych sekwencji lub do odkrywania nowych białek.

W pierwszym przypadku otrzymane widmo może zostać porównane z widmami znajdującymi się w bazie danych. Mocno upraszczając, widmo masowe każdej cząsteczki jest unikalne i metaforycznie można je traktować jako „odcisk palca" tego związku chemicznego.

W drugim przypadku, w celu poznania sekwencji aminokwasowej, bazuje się jedynie na otrzymanym widmie masowym. Z punktu widzenia tematyki tej rozprawy, ten

przypadek jest ciekawszy. Analiza widm masowych powstałych w różnych eksperymentach jest motywacją do stworzenia modeli matematycznych i sformułowania problemów kombinatorycznych.

W tym rozdziale zaprezentowano krótką klasyfikację problemów sekwencjonowania za pomocą spektrometrii mas. Klasyfikację tę wraz ze sformułowaniem problemów kombinatorycznych problemów kombinatorycznych autor rozprawy przedstawił w pracach [28, 29, 35]. W niniejszym rozdziale zaproponowano również wielomianowy algorytm sekwencjonowania de novo.

5.2 Metody

5.2.1 Degradacja Edmana

Degradacja Edmana [45] to reakcja izocyjanku fenylu z grupą -NH_2 N-terminalnego aminokwasu. Ta część reakcji zachodzi w środowisku zasadowym i jej produktem jest pochodna tiomocznika - karbamyl.

W kolejnym etapie karbamyl w środowisku kwaśnym przekształca się w związek cykliczny, a następnie rozpada do fenylotiohydantoinowej pochodnej aminokwasu (PTH-aminokwasu).

Wynikiem reakcji jest więc pochodna aminokwasu oraz łańcuch peptydowy skrócony o ten jeden (N-terminalny) aminokwas. Powstały PTH-aminokwas można następnie zidentyfikować w procesie elektroforezy.

Iteracyjne zastosowanie metody Edmana pozwala na poznanie kolejnych aminokwasów w cząsteczce. Ze względu na brak 100% wydajności degradacji Edmana może zostać wykorzystana do sekwencjonowania krótkich łańcuchów, które nie przekraczają 50 aminokwasów.

5.2.2 Spektrometria masowa

Gdy obojętna cząsteczka P ulega jonizacji tworzy się jon molekularny o dodatnim ładunku M^+. Jon ten może się rozpaść i utworzyć pewien jon fragmentacyjny A^+. Jon A^+ może się ponownie rozpaść tworząc kolejny jon fragmentacyjny B^+. Rozpady takie mogą zachodzić tak długo, dopóki jon zawiera wystarczająco dużo energii, aby się rozpaść. Ciąg takich rozpadów nazywany jest **drogą fragmentacji**.

Rozpad jonów fragmentacyjnych może zachodzić w różnych miejscach cząsteczki, może więc istnieć wiele różnych dróg fragmentacji dla analizowanego peptydu. Zbiór możliwych dróg fragmentacji nazywa się **schematem fragmentacji**. Schemat fragmentacji jest cechą charakterystyczną dla związku chemicznego.

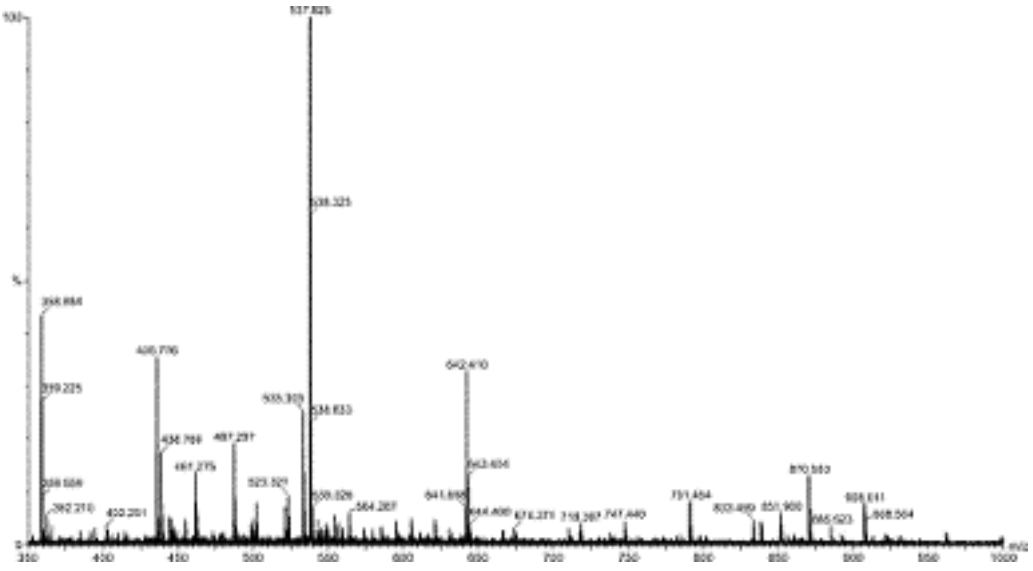
PRZYKŁAD: cztery przykładowe drogi fragmentacyjne dla hipotetycznej cząsteczki składającej się z 3 atomów (ABC):

- $ABC \rightarrow ABC^+$
- $ABC \rightarrow A^+ + BC$
- $ABC \rightarrow AB + C^+$
- $ABC \rightarrow AB^+ + C \rightarrow A^+ + B + C$

Jony powstałe w schemacie fragmentacji można rozdzielić wg stosunku ich masy do ładunku (m/e). Wykres, który na osi X przedstawia stosunek masy do ładunku (m/e) a

na osi Y stężenie odpowiednich jonów, w postaci tak zwanych **pików**, nazywa się **widmem masowym**. Przykładowe widmo tego rodzaju zostało przedstawione na rysunku 5.1.

Przebieg fragmentacji zależy od takich czynników jak dostarczona energia, czas przeprowadzania eksperymentu, własności fizyczne aparatu oraz czasu po jakim jony są rejestrowane. Dokładne odtworzenie tych samych warunków laboratoryjnych jest bardzo trudne, z tego powodu widma tego samego związku uzyskane w wyniku przeprowadzenia różnych eksperymentów mogą się nieco różnić.



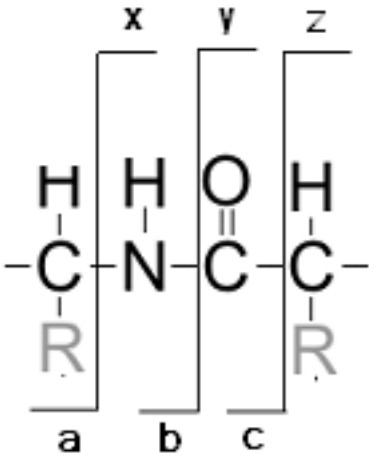
Rysunek 5.1: Przykład widma masowego

Urządzenie, które przeprowadza opisany eksperyment chemiczny nazywane jest spektrometrem masowym [30]. Składa się ono z następujących elementów:

- Układu wlotowy, który służy do wprowadzania próbki; jego budowa zależy zwykle od łatwości parowania badanych substancji; gazy i substancje bardzo lotne posiadają zimny wlot do urządzenia natomiast substancje, które trzeba ogrzać do wyższej temperatury w celu ich przejścia do stanu gazowego, posiadają wlot gorący, który można ogrzać do odpowiedniej temperatury.
- Źródła jonów (jonizatora), będącego miejscem, gdzie badana substancja jest jonizowana; istnieją różne metody jonizacji próbki: krótkofalowe promieniowanie o dużym natężeniu, przyśpieszone elektrony, pole elektryczne, jonizacja chemiczna lub laserowa.
- Analizatora, który służy do rozdzielenia jonów zgodnie z ich stosunkiem m/e; oddzielenie jonów wykonuje się przykładowo za pomocą pola elektrycznego, pola magnetycznego lub za pomocą badania czasu przelotu jonu.
- Detektora, który zlicza jony o danym stosunku m/e napływające z analizatora.
- Rejestratora danych (komputera), który na podstawie sygnału elektrycznego z detektora zapisuje widmo masowe.

W procesie fragmentacji jonu (molekularnego lub fragmentacyjnego) peptydu, następuje rozerwanie pewnego wiązania peptydowego, przy czym proces ten jest całkowicie losowy i nie można określić, które wiązanie peptydowe zostanie przerwane.

W wyniku rozerwania wiązania peptydowego w analizowanej cząsteczce ładunek dodatni może pozostać po N-końcowej albo po C-końcowej stronie łańcucha. Dodatkowo, można wyróżnić trzy miejsca w wiązaniu peptydowym, gdzie może zajść fragmentacja. W zależności od miejsca rozerwania cząsteczki oraz strony po której pozostał ładunek, wyróżnia się sześć typów jonów przedstawionych na rysunku 5.2. Na potrzeby dalszych rozważań te 6 typów jonów określane będą jako jony zewnętrzne.



Rysunek 5.2: Typy jonów podczas rozpadu wiązania peptydowego (źródło: rysunek własny)

Należy podkreślić, że próbka poddana eksperymentowi w spektrometrze składa się z wielu takich samych cząsteczek i dla każdej z nich następuje fragmentacja. Dla każdego jonu proces ten może zakończyć się w dowolnym momencie (fragment przestaje ulegać dalszemu rozpadowi). Zależy to od energii, która została dostarczona temu jonowi, czasu jaki trwa eksperyment i czynników losowych.

Teoretycznie, gdy próbka zawiera dostatecznie wiele cząsteczek oraz zapewnione zostały odpowiednie warunki eksperymentu tj. dostarczono odpowiednią ilość energii oraz przeznaczono na niego wystarczająco dużo czasu, to cząsteczki analizowanego peptydu powinny przejść wszystkie drogi fragmentacji. Dodatkowo powinny powstać wszystkie możliwe do uzyskania jony fragmentacyjne.

W rzeczywistym przypadku w trakcie eksperymentu mogą pojawić się błędy:

- **Błędem pozytywnym** określa się zarejestrowanie jonu, który nie wynika ze schematu fragmentacyjnego cząsteczki; taki jon może pochodzić z jonizacji zanieczyszczeń.

- **Błędem negatywnym** nazywa się niezarejestrowanie jonu, który należy do schematu fragmentacyjnego; zdarza się to, gdy podczas fragmentacji ten jon nie pojawił się lub w sytuacji, gdy uległ dalszej fragmentacji przed zarejestrowaniem.

W analizie wyników spektrometrii masowej peptydów pojawiają się dodatkowe utrudnienia, które należy uwzględnić, do których należą modyfikacje potranslacyjne oraz wyjątki.

Modyfikacje potranslacyjne białka to zmiany, które następują już po translacji. Mają one wpływ na własności fizyczne i chemiczne, aktywność cząsteczki jak również

na jej masę. Wpływają więc na postać widma cząsteczki. Znane modyfikacje zostały przedstawione w załączniku A-2.

Wyjątki są to pewne nietypowe sytuacje, które wpływają na masę jonów i które trzeba z tego względu wziąć pod uwagę analizując widmo. Przykładowo, w niektórych sytuacjach w trakcie fragmentacji z jonu może się wydzielić amoniak lub woda.

5.3 Klasyfikacja problemów sekwencjonowania za pomocą spektrometrii masowej

Można wyróżnić kilka podejść do problemu sekwencjonowania:

- przeszukiwanie baz danych
- markery sekwencji
- sekwencjonowanie de novo
- podejście mieszane

Przeszukiwanie baz danych polega na wyszukaniu w bazie widma, które odpowiada temu uzyskanemu w spektrometrze masowym. Jak wspomniano, bardzo trudne jest odtworzenie warunków eksperymentu i widmo tego samego związku, wykonane nawet przy tej samej konfiguracji sprzętu, może się nieco różnić. Algorytmy przeszukiwania baz danych powinny uwzględniać ten fakt. Przeszukanie bazy polega na znalezieniu najlepszego dopasowania. Popularne algorytmy przeszukujące bazy danych to m.in. Mascot [39], Tandem [18], Sequest[24]. Główną wadą tego podejścia jest to, że umożliwia ono rozpoznanie tylko znanych cząsteczek peptydów.

Podejście oparte na markerach sekwencji polega na wyborze ze spektrum tylko wybranych pików i poszukiwaniu w bazie danych sekwencji, które pasują do tych pików. Wynikiem przeszukania baz danych może być zbiór sekwencji, który powinien zostać poddany dodatkowej ocenie w celu znalezienia najlepszego dopasowania. Zasadniczą zaletą tego podejścia w stosunku do zwykłego przeszukiwania baz danych jest możliwość znalezienia sekwencji o nieznanych modyfikacjach potranslacyjnych. Dzięki rozpoznaniu kluczowych pików można odnaleźć w bazie danych peptyd przed modyfikacją potranslacyjną i przeprowadzić dodatkową analizę w celu stwierdzenia jaka modyfikacja się pojawiła. Podejście to nie pozwala na ustalenie budowy pierwszorzędowej nieznanych peptydów. Algorytmy, których działanie oparte jest na markerach sekwencji to m.in. GutenTag [20], SPIDER [49], OpenSea [44].

Metody sekwencjonowanie de novo bazują jedynie na widmie masowym i nie korzystają z dodatkowych informacji. Oczywistą zaletą takiego podejścia jest możliwość określenia sekwencji nieznanych aminokwasów.

Na potrzeby dalszych rozważań zostanie zdefiniowany pojedynczy pik z widma masowego jako uporządkowana dwójka (a, b), gdzie a to wartość na osi X dla tego (m/z), a b to wartość na osi Y (intensywność) dla tego piku. Widmo masowe W można przedstawić jako zbiór takich uporządkowanych dwójek.

Jeśli fragmentacji ulegają jedynie obojętne cząsteczki białek, to powstałe w ten sposób jony odpowiadają prefiksom albo sufiksom cząsteczki. Są to oczywiście jony wszystkich 6 typów, które zostały przedstawione na rysunku 5.2. Jeśli widmo masowe pewnej cząsteczki zawiera piki odpowiadające wszystkim jonom „a”, „b”, „c”, „x”, „y” oraz „z” możliwym do uzyskania z fragmentacji tej cząsteczki oraz nie zawiera żadnych innych jonów to nazywane jest widmem idealnym i oznaczone W^{abcxyz}.

Zostaną wyróżnione podzbiory widma idealnego, które zawierają wszystkie piki pochodzące od cięć jednego lub kilku wybranych typów i nie zawierają żadnych innych pików np. W^a zawiera wszystkie i tylko te piki, które pochodzą z jonów typu „a", natomiast $W^{a,x,z}$ zawiera wszystkie piki pochodzące od jonów „a", „x" oraz „z" i nie zawiera innych pików.

Jeśli jon fragmentacyjny ulega dalszej fragmentacji, to powstały po fragmentacji jon może odpowiadać wewnętrznemu fragmentowi cząsteczki. Takie jony nazywane są jonami wewnętrznymi. Widmo, które zawiera wszystkie możliwe do uzyskania piki, również te odpowiadające wewnętrznym fragmentom cząsteczki, zwane jest widmem pełnym i oznaczone W^*. Należy zauważyć, że $W^{abcxyz} \subseteq W^*$.

Ogólny problem sekwencjonowania de novo można sformułować następująco:

Problem 5.1.
Instancja:
Uporządkowana trójka (W,d,f), gdzie W to widmo masowe, m masa analizowanego peptydu a f to funkcja oceniająca trafność znalezionej sekwencji do widma masowego W.
Odpowiedź:
Sekwencja aminokwasowa o masie m maksymalizująca wartość funkcji f.

Złożoność obliczeniowa konkretnego problemu sekwencjonowania de novo zależy od dodatkowych czynników, takich jak błędy eksperymentalne, modyfikacje potranslacyjne cząsteczki i wyjątki. Istotnym czynnikiem, który decyduje o klasie złożoności problemu, jest zbiór typów jonów Δ.

Na wyróżnienie zasługuje przypadek, gdy w schemacie fragmentacji pojawiają się jedynie jony odpowiadające prefiksom i sufiksom sekwencji. Dzieje się tak, gdy obojętna cząsteczka peptydu ulega tylko jednokrotnemu zjonizowaniu i podziałowi (otrzymany jon nie przechodzi dalszej fragmentacji). Odpowiada to sytuacji, gdy do układu dostarczono relatywnie mało energii, niewystarczająco do przeprowadzenia dalszych podziałów. W literaturze przedstawiono kilka wielomianowych algorytmów rozwiązujących niektóre takie sytuacje. Większość metod bazuje na idei programowania dynamicznego. Poniżej omówiono kilka przypadków z takimi typami jonów.

W pierwszym przypadku fragmentacji podlega tylko obojętna cząsteczka i zawsze tworzy się jon jednego ustalonego typu. W rejestratorze odnotowano wszystkie możliwe jony fragmentacyjne. W tym wypadku drogę fragmentacyjną można przedstawić w postaci jednego równania:

$$P = M^+ + m$$

gdzie P jest cząsteczką peptydu, M^+ jest to dowolny jon ustalonego typu a m to pozostały, niezjonizowany fragment sekwencji. Dodatkowo zakłada się, że nie zarejestrowano jonów, które pochodziłyby z jonizacji zanieczyszczeń. Cząsteczka białka nie przeszła żadnych modyfikacji potranslacyjnych i nie wystąpiły żadne wyjątki.

Gdy jonizacji podlegał N-końcowy fragment cząsteczki (zachodziły cięcia typu „a" albo „b" albo „c") to na widmie pojawią się masy wszystkich N-końcowych jonów fragmentacyjnych. W takiej sytuacji różnica mas pomiędzy $k-tym$ a $(k-1)-szym$ pikiem odpowiada masie $k-tej$ reszty aminokwasowej licząc od N-końca. Natomiast gdy w eksperymencie zachodziły cięcia typu „x" albo „y" albo „z", to różnica pomiędzy $k-tym$ a $(k-1)$-szym pikiem odpowiada masie $k-tej$ reszty aminokwasowej licząc od C-końca. Przykładowo - różnica mas pomiędzy drugim a pierwszym pikiem odpowiada masie ostatniego aminokwasu w sekwencji. Przedstawionej sytuacji odpowiada następujący problem kombinatoryczny:

Problem 5.2.
Instancja:
Uporządkowana trójka (W,d,f), gdzie $W \in \{W^a, W^b, W^c, W^x, W^y, W^z\}$, m masa analizowanego peptydu a f to funkcja oceniająca trafność znalezionej sekwencji do widma masowego W.
Odpowiedź:
Sekwencja aminokwasowa o masie m maksymalizująca wartość funkcji f.

Rozwiązanie tego problemu polega na obliczeniu różnicy pomiędzy kolejnymi pikami, każda taka różnica determinuje kolejny (lub poprzedni) aminokwas w sekwencji. Problem ten można rozwiązać w czasie liniowym. Sytuacja taka jest jednak dość teoretyczna, gdyż trudno wymusić taki schemat fragmentacji cząsteczki.

W drugim przypadku w drodze fragmentacji tworzą się jony zewnętrzne odpowiadające fragmentom cząsteczki pochodzącym tylko z C-końca albo tylko z N-końca. Dodatkowo zakłada się, że powstają jony dwóch albo trzech typów, nie zarejestrowano zanieczyszczeń oraz powstają wszystkie możliwe jony fragmentacyjne spełniające te warunki. Problem kombinatoryczny związany z tą sytuacją można zdefiniować następująco:

Problem 5.3.
Instancja:
Uporządkowana trójka (W,d,f), gdzie $W \in \{W^{abc}, W^{ab}, W^{ac}, W^{bc}\}$, m masa analizowanego peptydu a f to funkcja oceniająca trafność znalezionej sekwencji do widma masowego W.
Odpowiedź:
Sekwencja aminokwasowa o masie m maksymalizująca wartość funkcji f.

Rozwiązanie problemu 3. można znaleźć przy wykorzystaniu podobnej metody jak dla problemu 2., rozpoznając wcześniej typy jonów po charakterystycznej różnicy mas pomiędzy odpowiadającymi im pikom (przykładowo: różnica mas pomiędzy jonem typu „c" a jonem typu „b" pochodzącymi z przerwania tego samego wiązania peptydowego jest stała i odpowiada masie grupy CO).

W wyniku fragmentacji powstać może tylko jon prefiksowy typu „b" oraz jon sufiksowy typu „y". Sytuacja ta odpowiada rzeczywistemu eksperymentowi spektrometrycznemu - dostarczając do układu niewielką energię, można w przybliżeniu uzyskać właśnie taką sytuację. Pokazano, że problemy kombinatoryczne związane z takim eksperymentem spektrometrycznym są łatwe obliczeniowo[47]:

- w przypadku idealnym (problem 4)
- w przypadku błędów pozytywnych (problem 5)
- w przypadku błędów negatywnych (problem 6)
- w przypadku, gdy występuje jedna modyfikacja potranslacyjna (problem 7)

Poniżej zdefiniowano problemy kombinatoryczne odpowiadające tym sytuacjom:

Problem 5.4.
Instancja:
Uporządkowana trójka (W,d,f), gdzie $W = W^{c,y}$, m masa analizowanego peptydu a f to funkcja oceniająca trafność znalezionej sekwencji do widma masowego W.
Odpowiedź:
Sekwencja aminokwasowa o masie m maksymalizująca wartość funkcji f.

Problem 5.5.
Instancja:

Uporządkowana trójka (W,d,f), gdzie $W^{c,y} \subseteq W$, m masa analizowanego peptydu a f to funkcja oceniająca trafność znalezionej sekwencji do widma masowego W.
Odpowiedź:
Sekwencja aminokwasowa o masie m maksymalizująca wartość funkcji f.

Problem 5.6.
Instancja:
Uporządkowana trójka (W,d,f), gdzie $W \subseteq W^{c,y}$, m masa analizowanego peptydu a f to funkcja oceniająca trafność znalezionej sekwencji do widma masowego W.
Odpowiedź:
Sekwencja aminokwasowa o masie m maksymalizująca wartość funkcji f.

Problem 5.7.
Instancja:
Uporządkowana trójka (W,d,f), gdzie $\underset{(a,b)\in W}{\forall}(a,b) \in W^{c,y} \vee (a-c,b) \in W^{c,y} \vee (a+c,b) \in W^{c,y}$ gdzie c oznacza masę modyfikacji potranslacyjnej , m masa analizowanego peptydu a f to funkcja oceniająca trafność znalezionej sekwencji do widma masowego W.
Odpowiedź:
Sekwencja aminokwasowa o masie m maksymalizująca wartość funkcji f.

Udowodniono, że problem sekwencjonowania z jonami wewnętrznymi, niezależnie od błędów eksperymentalnych, jest problemem trudnym obliczeniowo [16]. Problem ten można zdefiniować w następujący sposób:

Problem 5.8.
Instancja:
Uporządkowana trójka (W,d,f), gdzie $W \subseteq W^* \wedge W \not\subset W^{abcxyz}$, m masa analizowanego peptydu a f to funkcja oceniająca trafność znalezionej sekwencji do widma masowego W.
Odpowiedź:
Sekwencja aminokwasowa o masie m maksymalizująca wartość funkcji f.

Ostatnim sposobem ustalania sekwencji peptydów jest podejście mieszane. Łączy ono omówione w tym rozdziale trzy podejścia.

5.4 Algorytm sekwencjonowania peptydów de novo

W tym podrozdziale zostanie przedstawiony wielomianowy algorytm rozwiązujący problem sekwencjonowania de novo, gdy w widmie powstają 4 rodzaje jonów („a”, „c”, „y” oraz „z”). Dodatkowo zezwala się na występowanie błędów negatywnych jak i pozytywnych. Problem ten można zdefiniować następująco:

Problem 5.9.
Instancja:
Uporządkowana trójka (W,d), gdzie $X \subseteq W^{a,c,y,z}$, $X \subseteq W$, m masa analizowanego peptydu a W to widmo masowe.
Odpowiedź:
Sekwencja aminokwasowa o masie m do której pasuje maksymalnie wiele pików typu „c” z widma W. Jeśli istnieje wiele takich sekwencji to ta spośród nich do których pasuje możliwie dużo wszystkich plików z widma W.

Jak wspomniano, w literaturze można znaleźć kilka wielomianowych algorytmów do rozwiązania problemów, gdy w spektrum pojawiają się jony odpowiadające N-końcowym

i (lub) C-końcowym fragmentom cząsteczki. Należy zauważyć, że w przypadku występowania błędów negatywnych lub pozytywnych może zostać znalezionych wiele rozwiązań, które pasują do widma masowego. Założono, że znalezione rozwiązanie

- powinno uwzględniać maksymalną liczbę pików z widma masowego,
- powinno uwzględniać maksymalną liczbę tych pików, dla których istnieje możliwość weryfikacji typu.

Przedstawiony w tym rozdziale algorytm spełnia oba powyższe założenia. Poniżej uzasadniono słuszność tych założeń.

W przypadku bez błędów różnica pomiędzy sąsiadującymi pikami odpowiada masie pewnej reszty aminokwasowej. W przypadku błędów negatywnych, różnica mas pomiędzy dwoma pikami może odpowiadać masie kilku reszt aminokwasowych. Dodatkowo, jeśli dochodzą błędy pozytywne, może się zdarzyć, że jako część właściwego rozwiązania zostanie zakwalifikowane zanieczyszczenie, gdy różnica mas między odpowiadającym mu pikiem a innym pikiem pochodzącym z widma odpowiada masie jednej lub kilku reszt aminokwasowej. Takich sytuacji nie można wykluczyć i jednoznacznie stwierdzić, czy dany pik pochodzi z zanieczyszczenia, czy z jonu badanej cząsteczki. Jednakże bardziej prawdopodobna jest sytuacja, gdy ta różnica mas rzeczywiście odpowiada występowaniu tych reszt aminokwasowych w analizowanej cząsteczce. Przypadkowe dopasowanie zanieczyszczeń jest mało prawdopodobne. Przykładowo jeśli można znaleźć dwa rozwiązania problemu sekwencjonowania: jedno dopasowuje 8 pików a drugie 9 pików widma, to rozwiązanie zawierające 9 pików powinno być lepsze. Przedstawiona metoda będzie maksymalizować liczbę dopasowanych pików.

Dwa piki, pomiędzy którymi obliczana jest różnica mas, powinny odzwierciedlać jony fragmentacyjne z tej samej strony cząsteczki (oba prefiksy lub oba sufiksy sekwencji). Jeśli przykładowo zdarzy się sytuacja, że różnica pomiędzy pikiem jonu typu „a” oraz jonu typu „z” będzie odpowiadać masie pewnych reszt aminokwasowych, to wykorzystanie tej informacji przy dopasowaniu sekwencji jest oczywiście błędne, bo jony te pochodzą z przeciwnych końców cząsteczki. Jedynie różnica mas pomiędzy jonami pochodzącymi z tej samej strony cząsteczki może poprawnie identyfikować reszty aminokwasowe w sekwencjonowanej cząsteczce.

Zaproponowane podejście jest następujące: skoro nie można stwierdzić, które piki odpowiadają którym typom jonów, to wszystkie jony zostaną potraktowane jako jony typu „c” - tych jonów w doświadczeniu jest najwięcej. Odtwarzając sekwencję można jednak rozpoznać niektóre jony typu „c”. Jeśli w jednej cząsteczce nastąpiło przerwanie pewnego wiązania peptydowego i utworzył się jon typu „c”, a w drugiej cząsteczce przerwano to same wiązanie i utworzył się jon typu „a”, to na widmie pojawią się piki, których różnica mas odpowiada masie grupy NHCO. O masę tej grupy różnią się bowiem oba jony (5.2). Fakt ten można wykorzystać do dodatkowej weryfikacji jonów typu „c”: są to piki, dla których istnieją w widmie piki lżejsze o masę grupy NHCO. Informacja jest dostępna tylko dla niektórych pików. Rozwiązania maksymalizujące liczbę „zweryfikowanych” jonów typu „c” powinny minimalizować liczbę pomyłek: zakwalifikowania do rozwiązania zanieczyszczeń i jonów innego typu. Dodatkowo założono, że drugie założenie jest silniejsze, gdyż prowadzi do maksymalizacji w rozwiązaniu liczby jonów odpowiedniego typu. Przedstawiony w tym podrozdziale algorytm:

- zwraca jako rozwiązanie cząsteczkę, która zawiera jak najwięcej jonów typu „c”,
- jeśli istnieje wiele takich rozwiązań, to algorytm wybierze to o maksymalnej liczbie wszystkich jonów.

Widmo masowe można przedstawić w postaci grafu skierowanego $G = (V, A)$, gdzie każdy pik odpowiada wierzchołkowi tego grafu. W grafie istnieje łuk $(v_i, v_j) \in A$ wtedy i tylko wtedy, gdy różnica mas pomiędzy pikiem odpowiadającym wierzchołkowi v_j a pikiem odpowiadającym wierzchołkowi v_i jest równa masie dowolnych k reszt aminokwasowych. Wagą tego łuku jest k. Tak zdefiniowany graf jest acyklicznym grafem skierowanym.

Z każdym wierzchołkiem v w grafie G powiązana zostanie wartość $m(v)$ będąca względną masą jonu uzyskaną z widma masowego. Jeśli istnieje łuk $(v_i, v_j) \in A$, to $m(v_j) - m(v_i) = C_k$, gdzie C_k jest sumaryczną masą k reszt aminokwasowych. Aby znaleźć k reszt aminokwasowych odpowiadających masie C_k można skorzystać z następującej zależności rekurencyjnej:

$$C_k = C_{k-1} + R_x$$

oraz

$$C_1 = R_x$$

gdzie R_x jest masą dowolnej reszty aminokwasowej.

Aby zdecydować, czy istnieje łuk $(v_i, v_j) \in A$, należy znaleźć rozwiązanie powyższego równania rekurencyjnego. W tym celu można rozwiązać poniższy wariant decyzyjnego problemu plecakowego (problem 10). Problem ten został nazwany **wielomianowym decyzyjnym problemem plecakowym**.

Problem 5.10.
Instancja:

- pojemność plecaka $b = m(v_j) - m(v_i)$,
- zbiór elementów $Z = \{R_{1,1}, R_{1,2}, ..., R_{1,k_1}, R_{2,1}, R_{2,2}, ..., R_{2,k_2}, ..., R_{20,1}, R_{20,2}, ..., R_{20,k_{20}}\}$, gdzie $R_{x,y}$ odpowiada masie reszty aminokwasowej R_x,
- rozmiar $s(R_{x,y})$ każdego elementu $R_{x,y} \in Z$ spełniający równanie $s(R_{x,y}) = R_x$,
- wartość $w(R_{x,y})$ każdego elementu $R_{x,y} \in Z$ spełniający równanie $w(R_{x,y}) = R_x$.

Odpowiedź:
Czy istnieje podzbiór $Z' \subseteq Z$, taki że:

$$\sum\nolimits_{R_{x,y} \in Z'} s(R_{x,y}) = b$$

oraz:

$$\sum\nolimits_{R_{x,y} \in A'} w(R_{x,y}) = b$$

Komentarza wymaga sposób tworzenia zbioru Z. Element $R_{x,y}$ odpowiada reszcie aminokwasowej R_x. Należy zauważyć, że tak zdefiniowany plecak może zostać wypełniony kilkoma elementami o takim samym rozmiarze i wartości - co odpowiada sytuacji, gdy pomiędzy dwoma pikami na widmie istnieje kilka takich samych reszt aminokwasowych. W celu zamodelowania tej sytuacji wprowadzono kopie elementów, które opisują tą samą resztę aminokwasową. Liczbę elementów k_x odpowiadających reszcie aminokwasowej R_x można obliczyć z poniższego wzoru:

$$k_x = \left\lfloor \frac{C_k}{R_x} \right\rfloor$$

W ogólności, złożoność problemu plecakowego wynosi $O(a \cdot b)$ gdzie a to liczba elementów a b to pojemność plecaka. W zdefiniowanym tutaj problemie pojemność plecaka

oraz liczba przedmiotów są ograniczone przez wartość $C_k = m(v_j) - m(v_i)$. Złożoność tego wariantu problemu plecakowego wynosi więc $O({C_k}^2)$. Należy zauważyć, że skoro największa liczba w instancji tej wersji plecaka to C_k, dodatkowo liczba elementów

$$|Z| = \lfloor \frac{C_k}{R_1} \rfloor + \lfloor \frac{C_k}{R_2} \rfloor + ... + \lfloor \frac{C_k}{R_20} \rfloor$$

jest ograniczona wielomianowo od C_k to przedstawiony algorytm jest wielomianowy.

Pokazano, że w czasie $O({C_k}^2)$ można zweryfikować, czy między dwoma wierzchołkami grafu istnieje łuk. Należy zauważyć, że C_k można ograniczyć odgórnie stałą m, gdzie m jest masą całej cząsteczki. Sprawdzenie istnienia łuków pomiędzy każdą parą wierzchołków w grafie można więc wykonać w czasie $O(n^2 \cdot m)$, gdzie $n = |V|$. Należy zauważyć, że liczba wierzchołków grafu n, która odpowiada liczbie pików w widmie jest liniowo zależna od masy cząsteczki m, stąd ostatecznie złożoność obliczeniowa tego kroku algorytmu to $O(m^4)$. Skoro liczba elementów w instancji n jak i największa liczba w instancji $max(C_k)$ są ograniczone wielomianowo od m, to sprawdzenie istnienia łuków pomiędzy każdą parą wierzchołków w tym grafie można przeprowadzić w czasie wielomianowym.

Ten krok algorytmu można znacząco zoptymalizować. W tym celu wszystkie problemy plecakowe, jakie należy rozwiązać, wystarczy posortować rosnąco względem b. Rozwiązując kolejny problem można skorzystać z tablicy poprzedniego problemu. Sprowadza się to do rozwiązania jednego problemu plecakowego, dla najwyższej wartości C_k. Złożoność tego kroku została więc zredukowana do $O(m^2)$.

Przedstawiono w tym podrozdziale wariant problemu plecakowego należy uogólnić, aby uwzględnić modyfikacje i wyjątki. Każda modyfikacja i wyjątek wymagają indywidualnego podejścia. W ogólności, wiele takich przypadków można zamodelować powiększając zbiór Z o dodatkowe elementy odpowiadające masie zmodyfikowanych reszt aminokwasowych. Ze względu na potrzebę indywidualnego podejścia do każdej modyfikacji i wyjątku, w tej pracy przedstawiono jedynie ogólną postać **wielomianowego decyzyjnego problemu plecakowego**.

Waga wszystkich łuków wchodzących do wierzchołków, które odpowiadają zweryfikowanym jonom typu „c", zostaje obniżona o 1. Można zatem zauważyć, że w przypadku idealnym, gdy widmo zawiera piki wszystkich jonów, to wierzchołki odpowiadające pikom typu „c" jako jedyne będą miały wszystkie łuki wchodzące o wadze równej 0.

W tak zmodyfikowanym grafie należy znaleźć najkrótszą skierowaną ścieżkę zawierającą dodatkowo możliwie największą liczbę wierzchołków. Ścieżka ta jest związana z sekwencjami aminokwasów (gdyż może być ich wiele), które stanowią rozwiązanie problemu. Najkrótsza ścieżka gwarantuje wykorzystanie do znalezienia rozwiązania maksymalnie wiele pików z widma o których wiadomo, że są pikami typu „c". Wybór tej spośród najkrótszych ścieżek, która zawiera najwięcej wierzchołków gwarantuje wykorzystanie do rozwiązania maksymalnie wiele pików z widma.

Do rozwiązania problemu proponuje się lekko zmodyfikowany algorytm Dijkstry znajdowania najkrótszej skierowanej ścieżki w grafie:

1. Utworzenie tablicy odległości. Odległość do źródła„s" wynosi 0. Odległość do wszystkich pozostałych wierzchołków wynosi nieskończoność. Utworzenie zbioru zawierającego nieodwiedzone wierzchołki. Zbiór ten zawiera wszystkie wierzchołki grafu. Utworzenie tablicy poprzedników. Jako poprzednik każdego wierzchołka ustawia się wartość „brak poprzednika".

2. Jako wierzchołek aktualny przyjmuje się wierzchołek „s".

3. Obliczenie tymczasowych odległości do wszystkich nieodwiedzonych bezpośrednich następników aktualnego wierzchołka. Dla każdego wierzchołka, dla którego obliczona odległość tymczasowa jest mniejsza lub równa niż dotychczas zapisana odległość, za nową odległość do tego wierzchołka przyjmuje się obliczoną odległość tymczasową.

4. Usunięcie wierzchołka aktualnego z listy nieodwiedzonych wierzchołków.

5. Jako wierzchołek aktualny przyjmuje się nieodwiedzony wierzchołek, do którego obliczona odległość tymczasowa jest najmniejsza. Jako poprzednika nowego wierzchołka aktualnego przyjmuje się poprzedni wierzchołek aktualny.

6. Jeśli zbiór nieodwiedzonych wierzchołków jest pusty, należy zakończyć algorytm. W przeciwnym wypadku jako wierzchołek aktualny ustawia się ten spośród nieodwiedzonych wierzchołków, dla którego obliczona odległość jest najmniejsza. Należy przejść do punktu 3. algorytmu.

Komentarza wymaga sposób wyboru i podejście do wierzchołków „s" oraz „t":

- Wierzchołek „t" odpowiada pikowi o najwyższej masie, który reprezentuje zjonizowaną cząsteczkę; $m(t)$ zostało pomniejszone o masę grupy funkcyjnej -COOH, aby dla dowolnego wierzchołka v opisującego jon typu „c" różnica $m(t) - m(v)$ odpowiadała masie reszt aminokwasowych bez grupy funkcyjnej.

- Wierzchołek „s" odpowiada masie grupy NH_2, aby dla dowolnego wierzchołka v opisującego jon typu „c" różnica $m(v) - m(s)$ odpowiada masie reszt aminokwasowych bez grupy funkcyjnej. Jest to sztuczny wierzchołek niezwiązany z żadnym pikiem.

Złożoność tego kroku algorytmu zależy od implementacji algorytmu Dijkstry i wynosi $O(n^2)$ w przypadku, gdy zbiór potencjalnych wierzchołków do wyboru jest zaimplementowany w postaci listy. W przypadku wykorzystania kopca zamiast listy złożoność algorytmu można poprawić uzyskując $O(n \cdot log(n))$ [42]. Złożoność obliczeniową tego kroku algorytmu można ograniczyć $O(m^2)$.

Aby otrzymać ścieżkę wierzchołków tworzących rozwiązanie, to rozpoczynając od wierzchołka „t" należy poruszać się do poprzednika, aż do osiągnięcia wierzchołka „s". Złożoność tego kroku algorytmu wynosi $O(n)$ i można ją ograniczyć odgórnie $O(m)$.

Podsumowując rozważania, złożoność całego algorytmu wynosi $O(m^2)$, gdzie m jest masą analizowanej cząsteczki. Należy przypomnieć, że instancją problemu jest widmo masowe W oraz masa peptydu m. W można przedstawić jako zbiór wierzchołków V grafu G. Liczba tych wierzchołków jest ograniczona wielomianowo od m a największa liczba w instancji to m, przedstawiony algorytm jest więc wielomianowy.

5.5 Wnioski

W rozdziale przestawiono metody ustalania krótkich sekwencji łańcuchów peptydowych.

Pierwsza metoda to degradacja Edmana. Jest to metoda analityczna, która pozwala na ustalenie kolejności do 50 aminokwasów w cząsteczce. Wadą tego podejścia jest to, że wymaga relatywnie dużej ilości oczyszczonego białka do eksperymentu.

Nowszą i skuteczniejszą metodą jest spektrometria masowa. Wynikiem eksperymentu w spektrometrze jest tzw. widmo masowe, które jest charakterystyczne dla badanej cząsteczki. W celu ustalenia sekwencji znanego białka, bazując na jego widmie masowym,

można przeszukać bazę danych w celu dopasowania widm. W przypadku nowego białka stosuje się metody de novo.

W przypadku ogólnym, dla dowolnego schematu fragmentacji cząsteczki, problem sekwencjonowania de novo jest trudny obliczeniowo. Znane są jednak wielomianowe algorytmy dla niektórych wersji problemów, gdy w wyniku fragmentacji powstają jedynie jony odpowiadające prefiksom lub (oraz) sufiksom cząsteczki. Wkładem autora w dziedzinę jest wielomianowy algorytm de novo, który rozwiązuje jeden z problemów sekwencjonowania.

Zarówno degradacja Edmana jak i spektrometria masowa pozwalają jedynie na poznanie krótkich sekwencji. Potrzebne są metody obliczeniowe, które pozwolą na ustalenie kolejności aminokwasów w długich cząsteczkach. Zostaną one opisane w kolejnym rozdziale pracy.

Rozdział 6

Asemblacja łańcuchów peptydowych

6.1 Wstęp

Brak bezpośrednich metod ustalania sekwencji długich łańcuchów peptydowych powoduje, że potrzebne są metody asemblacyjne, które pozwolą poskładać krótkie łańcuchy peptydowe w większą całość. W poniższym rozdziale rozważa się eksperyment biochemiczny, którego wyniki posłużą do zdefiniowania problemów asemblacji łańcuchów peptydowych.

W rozdziale tym omawia się przypadek idealny, tj. gdy w eksperymencie nie pojawiają się żadne błędy. Sytuacja taka jest jednak mało prawdopodobna. W literaturze zdefiniowano model grafowy, który opisuje wyniki takiego eksperymentu[6][26]. Przedstawiony matematyczny model został formalnie w tym rozdziale zdefiniowany oraz pokazano metodę rozpoznawania zaproponowanych grafów. W rozdziale określono złożoność obliczeniową problemu asemblacji peptydów w przypadku bez błędów, gdy dany jest dodatkowo rozkład aminokwasów.

W rozdziale niniejszym rozważa się również przypadek z błędami wynikającymi z eksperymentu biochemicznego. Proponuje się model grafowy do przedstawienia wyników tej wersji eksperymentu biochemicznego. W literaturze udowodniono, że rozważany problem z błędami jest silnie NP-trudny [27]. Przeprowadzono badania mające na celu zaprojektowanie i przetestowanie różnych podejść metaheurystycznych dla tego problemu. W rozdziale tym zaprezentowano dwa nowe podejścia metaheurystyczne i przeprowadzono porównanie tych metod z metodą Tabu opisaną w literaturze. W rozdzialem tym zaproponowano heurystykę dedykowaną, która daje bardzo dobre wyniki.

Ponadto przedstawiono tutaj klasyfikację problemów asemblacji.

Warto zaznaczyć, że wszystkie rozważania dotyczące grafów, które zostały przedstawione w tym rozdziale, dotyczą grafów skierowanych.

6.2 Eksperyment biochemiczny

Wynikiem sekwencjonowania jest poznanie jedynie krótkich łańcuchów peptydowych. Naturalnym podejściem, które pozwoli na wyznaczenie dłuższych sekwencji, jest pocięcie długich łańcuchów na krótsze i zsekwencjonowanie tych krótkich łańcuchów. Problemem, który powstaje przy takim podejściu, jest odtworzenie kolejności krótkich łańcuchów w oryginalnej cząsteczce.

W celu uzyskania krótkich łańcuchów aminokwasów i otrzymania informacji, która pozwoli odtworzyć ich kolejność w oryginalnej cząsteczce proponuje się eksperyment biochemiczny z wykorzystaniem endopeptydaz [31].

Endopeptydaza jest związkiem, który tnie łańcuch peptydowy w dobrze określonym miejscu - po wystąpieniu w łańcuchu konkretnych aminokwasów. Endopeptydazę można więc zdefiniować przez zbiór aminokwasów, które rozpoznaje i po których przeprowadza cięcie. Na potrzeby tego eksperymentu zaproponowano wykorzystanie dwóch endopeptydaz, dla których zbiory aminokwasów po których przeprowadzają cięcie są rozłączne. Badany peptyd rozdziela się do dwóch naczyń. W każdym z naczyń przeprowadza się reakcję z inną peptydazą. W doświadczeniu można przykładowo wykorzystać trypsynę i chymotrypsynę:

- trypsyna tnie łańcuch aminokwasów po wystąpieniu argininy lub lizyny;
- chymotrypsyna rozpoznaje natomiast tryptofan, fenyloanilinę i tyrozynę.

Otrzymaną mieszaninę krótkich peptydów rozdziela się wykorzystując do elektroferezę, a następnie sekwencjonuje się krótkie łańcuchy.

W reakcji trawienia łańcucha peptydowego endopeptydazą miejsca w łańcuchu, gdzie powinny zajść cięcia są dokładnie określone. Przypadek idealny to sytuacja, gdy wszystkie oczekiwane cięcia zachodzą.

Eksperyment chemiczny może być źródłem różnego rodzaju błędów. Tradycyjnie w problemach rozpoznawania sekwencji biologicznych wprowadza się klasyfikację błędów na pozytywne oraz negatywne.

Błędem pozytywnym wystąpienie łańcucha, który nie jest rzeczywistym wynikiem eksperymentu biologicznego lub biochemicznego. Może on powstać w wyniku przekłamania metody odczytującej krótkie sekwencje lub w wyniku zanieczyszczeń badanej próbki.

Błędem negatywnym jest niewystąpienie w zbiorze sekwencji, która powinna. Taka sytuacja może mieć miejsce, gdy pewna krótka sekwencja została zagubiona. Źródłem błędów negatywnych może być również brak informacji o powtórzeniach krótkich łańcuchów (w przypadku, gdy analiza chemiczna mieszaniny nie pozwala na otrzymanie takiej informacji).

Źródłem błędów w zaproponowanym eksperymencie biochemicznym jest proces trawienia. Błędem jest sytuacja, gdy nie zachodzą wszystkie cięcia, które wynikałyby z mechanizmu działania endopeptydazy. Taka sytuacja zdarza się w praktyce dość często.

Istnieje możliwość poznania rozkładu aminokwasów w badanej cząsteczce. W tym celu można przeprowadzić całkowite trawienie cząsteczki do aminokwasów a następnie zbadać stężenie tych aminokwasów w roztworze. Na tej podstawie można ustalić rozkład aminokwasów w oryginalnym peptydzie. W pracy rozważa się zarówno problemy, gdy ta dodatkowa informacja o rozkładzie aminokwasów jest dana oraz przypadki, gdy rozkład jest nieznany.

6.3 Przypadek idealny

Przypadek idealny to sytuacja, gdy zachodzą wszystkie cięcia w procesie trawienia endopeptydazami [27][6][26]. Dodatkowo w przypadku idealnym zakłada brak błędów negatywnych wynikających z powtórzeń. Rysunek 6.1 przedstawia przykładowy eksperyment asemblacji dla przypadku idealnego.

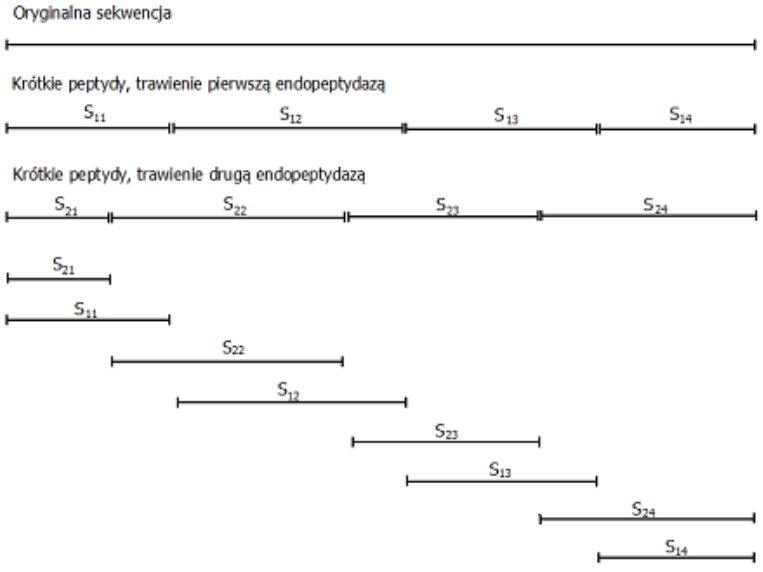
Zauważmy, że po przeprowadzeniu eksperymentu biochemicznego istnieje możliwość weryfikacji, czy eksperyment był pozbawiony błędów negatywnych (np. wynikających z braku informacji o powtórzeniach). W tym celu można przeprowadzić całkowite trawienie, aby ustalić liczbę wystąpień aminokwasów (rozkład) w szukanej cząsteczce i

porównać go z sumaryczną liczbą wystąpień aminokwasów w krótkich peptydach w obu naczyniach. Jeśli rozkłady te są równe, to eksperyment biochemiczny jest wolny od błędów negatywnych.

Pewną informację czy w wyniku eksperymentu biochemicznego pojawiły się błędy negatywne daje porównanie rozkładu aminokwasów w krótkich peptydach w pierwszym naczyniu z rozkładem aminokwasów w krótkich peptydach w drugim naczyniu. Informacja ta jednak nie jest kompletna. W przypadku, gdy te rozkłady różnią się, to jest to informacja, że pojawiły się błędy negatywne. W przypadku, gdy rozkłady aminokwasów są takie same to nie ma jednak pewności, że nie pojawiły się błędy negatywne. Błędy te mogły się pojawić, jednak rozkład aminokwasów z zagubionych peptydów w pierwszym i drugim naczyniu był taki sam.

Jak wspomniano, w rzeczywistym eksperymencie biochemicznym sytuacja idealna jest mało prawdopodobna. Stanowi jednak dobry punkt wyjściowy w badaniach metod asemblacyjnych.

Wynikiem eksperymentu biochemicznego są dwa zbiory krótkich peptydów. Na potrzebę eksperymentu wykorzystano enzymy tnące sekwencje w różnych miejscach, dzięki temu udało się zachować pewną informację o kolejności krótkich łańcuchów w oryginalnej cząsteczce. Częściowym nakładaniem łańcuchów nazywamy sytuację, gdy ciąg aminokwasów, którym kończy się jedna krótka sekwencja, jest taki sam jak ciąg aminokwasów, którym zaczyna się kolejna sekwencja. W celu odtworzenia oryginalnej cząsteczki należy znaleźć taką permutację łańcuchów, gdzie każda kolejna sekwencja częściowo pokrywa się z poprzednią. Specyfika eksperymentu wymaga dodatkowo, aby każde dwa kolejne łańcuchy w permutacji pochodziły z eksperymentu z inną endopeptydazą.



Rysunek 6.1: Asemblacja (przypadek bez błędów) - rysunek poglądowy (rysunek własny)

W pracach [26][6] zdefiniowano graf skierowany $G = (V, A)$ opisujący wyniki zaproponowanego eksperymentu.

Każdemu aminokwasowi przyporządkowuje się jedną literę. Dla porządku i przejrzystości rozważań przedstawionych w tej pracy proponuje się wykorzystanie formatu FASTA przedstawionego w załączniku A.3. Niech alfabet Σ będzie zbiorem symboli reprezentujących wszystkie aminokwasy białkowe ($|\Sigma| = 20$). Krótkie łańcuchy aminokwasów

można przedstawić jako słowa zbudowane nad tym alfabetem.

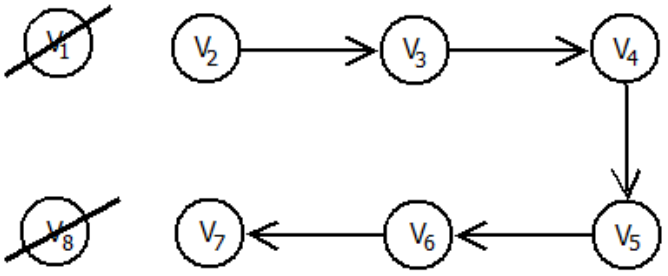
Każdemu słowu odpowiada wierzchołek tego grafu. Graf G jest grafem dwudzielny, wierzchołki dzieli się na dwa zbiory V_1 oraz V_2 i każdy zbiór zawiera wierzchołki odpowiadające sekwencjom otrzymanym w reakcji z inną endopeptydazą. Wierzchołki są zaetykietowane przez związane z nimi słowa.

Pomiędzy parą wierzchołków pochodzących z różnych zbiorów istnieje łuk, gdy sufiks etykiety poprzednika jest równy prefiksowi etykiety następnika. Długość tego prefiksu (sufiksu) jest wagą odpowiadającego mu łuku. Odnosząc się do eksperymentu biochemicznego, długość łuku determinuje wzajemne położenie dwóch sąsiednich krótkich peptydów w odtwarzanej cząsteczce.

Rozwiązanie problemu asemblacji jest permutacja krótkich łańcuchów, w naturalny sposób przekłada się na permutację wierzchołków w grafie. Każde dwa sąsiadujące wierzchołki należą do różnych zbiorów oraz sufiks etykiety wierzchołka stojącego na wcześniejszym miejscu w permutacji jest taki sam jak prefiks etykiety wierzchołka stojącego bezpośrednio za nim. Skoro każda kolejna krótka sekwencja w rozwiązaniu częściowo pokrywa się z poprzednią, to jedynie niezerowe nałożenia pomiędzy etykietami sąsiadujących wierzchołków mają uzasadnienie biologiczne. Pewien wierzchołek v_i może poprzedzać w rozwiązaniu wierzchołek v_j, wtedy i tylko wtedy, gdy $(v_i, v_j) \in A$.

Ze zbiorów V_1 oraz V_2 można usunąć wszystkie te wierzchołki, których etykieta całkowicie zawierają się w dowolnej etykiecie wierzchołka z drugiego zbioru. Takie wierzchołki nie wnoszą żadnych informacji do rozwiązania. Interesujące z perspektywy informacji o budowie oryginalnego peptydu, są jedynie wierzchołki, których etykieta tylko częściowo pokrywa się z etykietą innego wierzchołka. Fragment etykiety wykraczający poza to pokrycie dostarcza informacje o kolejnym fragmencie sekwencji w oryginalnym łańcuchu.

Na rysunku 6.2 przedstawiono graf dla sekwencji z rysunku 6.1. Wierzchołki od v_1 do v_8 odpowiadają kolejno sekwencjom: S_{21}, S_{11}, S_{22}, S_{12}, S_{23}, S_{13}, S_{24}, S_{14}.



Rysunek 6.2: Graf dla przypadku bez błędów - rysunek poglądowy

Zdefiniowany w ten sposób graf jest więc **grafem sprzężonym**.

Dla dowolnego łuku (v_i, v_j) w grafie G, gdzie $(v_i \in V_1 \wedge v_j \in V_2) \vee (v_i \in V_2 \wedge v_j \in V_1)$, waga łuku spełnia równanie:

$$w : A \rightarrow NN, w(v_i, v_j) = p$$

gdzie p spełnia następujący warunek:

$$\forall_{q \in \{1,2,\ldots,p\}} s_i(k-1+q) = s_j(q)$$

dodatkowo musi być spełniony warunek:

$$s_i(k-1) \in (C_1 \cup C_2)$$

gdzie $s_i(q)$ oznacza q-tą literę etykiety wierzchołka v_i, C_1 to zbiór liter odpowiadający aminokwasom po których następuje cięcie w pierwszym eksperymencie, natomiast C_2 to zbiór liter odpowiadający aminokwasom po których mogło zajść cięcie w drugim eksperymencie. Litera poprzedzająca nałożenie odpowiada aminokwasowi, po którym mogło zajść cięcie.

Powyższy warunek można wykorzystać do sprawdzenia, czy istnieje łuk pomiędzy wskazaną parą wierzchołków, a co za tym idzie do utworzenia wszystkich łuków bazując na danym zbiorze wierzchołków.

Poniżej przedstawiono metodę budowania grafu w oparciu o wyniki eksperymentu obliczeniowego:

METODA BUDOWANIA GRAFU PEPTYDOWEGO $G = (V, A)$

1. Utworzenie zbioru wierzchołków V_1, każdy wierzchołek odpowiada krótkiej sekwencji powstałej z trawienia długiego peptydu pierwszą endopeptydazą

2. Utworzenie zbioru wierzchołków V_2, każdy wierzchołek odpowiada krótkiej sekwencji powstałej z trawienia długiego peptydu drugą endopeptydazą

3. Każdemu wierzchołkowi $v \in V$, gdzie $V = V_1 \cup V_2$, zostaje przyporządkowana etykieta $\epsilon(v)$. Etykieta wierzchołka jest słowem nad alfabetem Σ. Słowo to reprezentuje peptyd powiązany z tym wierzchołkiem.

4. Utworzenie łuków (v_i, v_j), gdzie $(v_i \in V_1 \wedge v_j \in V_2) \vee (v_i \in V_2 \wedge v_j \in V_1)$ oraz $\epsilon(v_i)$ zakończona jest literą ze zioru C_1 (gdy $v_i \in V_1$) lub literą ze zbioru C_2 (w przeciwnym wypadku). Warunkiem utworzenia łuku jest istnienie takiego $p > 0$, że spełniony jest warunek: $\forall_{q\in\{1,2,...,p\}}\, s_i(k-1+q) = s_j(q)$. Dodatkowo wymagane jest aby $s_i(k-1) \in (C_1 \cup C_2)$. Łukowi przyporządkowuje się wagę p.

Skonstruowany za pomocą zaproponowanego algorytmu graf jest dwudzielny i sprzężony. Klasę grafów, która przedstawia wyniki eksperymentu obliczeniowego, nazwano **grafami peptydowymi**. Poniżej formalna definicja tej klasy grafów:

Def. 6.3.1. Niech Σ będzie alfabetem takim, że $|\Sigma| = 20$. Niech C_1 i C_2 będą rozłącznymi podzbiorami Σ, takimi że $|C_1 \cup C_2| < 20$. Dwudzielny Graf sprzężony $G = (V, A)$ nazywamy grafem peptydowym, jeśli możliwe jest przypisanie każdemu wierzchołkowi v tego grafu etykiety $\epsilon(v) = ((l_1(v), l_2(v), ..., l_{k(v)}(v))$ długości $k(v)$ i takiej, że:

1. $k(v) \geqslant 2$

2. $\forall_{v\in V}\forall_{i\in\{1,2,..,k(v)\}} l_i(v) \in \Sigma$

3. $(v_1, v_2) \in A \Leftrightarrow \exists_{p>0\wedge p<\min\{k(v_1),k(v_2)\}}\forall_{i\in\{1,2,...,p\}} l_i(v_2) = l_{k(v_1)-p+i}(v_1)$

4. $(\exists_{i<k(v_1)} l_i(v_i) \in C_1 \wedge l_{k(v_1)} \in C_2) \vee (\exists_{i<k(v_1)} l_i(v_i) \in C_2 \wedge l_{k(v_1)} \in C_1)$

Rozwiązaniem problemu asemblacji bez błędów w grafie peptydowym $G = (V, A)$ jest skierowana ścieżka przechodząca przez każdy wierzchołek tego grafu dokładnie raz, czyli skierowana ścieżka Hamiltona w grafie G. Pokazano, że graf ten jest grafem sprzężonym [26][6]. Problem znalezienia skierowanej ścieżki Hamiltona w grafie sprzężonym można sprowadzić do znalezienia skierowanej drogi Eulera w innym grafie. Znane są wielomianowe algorytmy znajdowania skierowanej drogi Eulera, więc rozważany problem jest łatwy obliczeniowo.

Zostanie wykazane, że grafy peptydowe są izomorficzne z klasą dwudzielnych grafów sprzężonych. Pokazano, że każdy graf peptydowy jest dwudzielnym grafem sprzężonym

[26][6]. Należy jeszcze pokazać, że każdy dwudzielny graf sprzężony może zostać zaetykietowany w taki sposób, aby spełniał wymogi formalnej definicji grafów peptydowych.

Na potrzeby dowodu zostanie wprowadzona pomocnicza klasa uproszczonych grafów peptydowych:

Def. 6.3.2. Niech Σ będzie alfabetem takim, że $|\Sigma| = 3$. Niech C_1 i C_2 będą rozłącznymi podzbiorami Σ, takimi że $|C_1| = 1$ oraz $|C_2| = 1$. Dwudzielny graf sprzężony $G = (V, A)$ nazywamy uproszczonym grafem peptydowym, jeśli możliwe jest przypisanie każdemu wierzchołkowi v tego grafu etykiety $\epsilon(v) = ((l_1(v), l_2(v), ..., l_k(v))$ (gdzie $k(v)$ to długość etykiety $\epsilon(v)$), takiej że:

1. $k(v) \geqslant 2$
2. $\forall_{v \in V} \forall_{i \in \{1,2,..,k(v)\}} l_i(v) \in \Sigma$
3. $(v_1, v_2) \in A \Leftrightarrow \exists_{p>0 \wedge p<\min\{k(v_1),k(v_2)\}} \forall_{i \in \{1,2,...,p\}} l_i(v_2) = l_{k(v_1)-p+i}(v_1)$
4. $(\exists_{i<k(v_1)} l_i(v_i) \in C_1 \wedge l_{k(v_1)} \in C_2) \vee (\exists_{i<k(v_1)} l_i(v_i) \in C_2 \wedge l_{k(v_1)} \in C_1)$

Poniżej podano algorytm, który przyporządkowuje wierzchołkom dwudzielnego grafu sprzężonego etykiety w taki sposób, że powstały graf jest uproszczonym grafem peptydowym:

1. Wierzchołki grafu G zostają rozdzielone na podzbiory $p_1 \cup p_2 \cup ... \cup p_n = V$, takie że $p_i \cap p_j = \emptyset$ dla każdego $i \neq j$. Do każdego zbioru należą tylko te wierzchołki, których lista następników jest taka sama.
2. Każdemu zbiorowi p_i przyporządkowuję się liczbę i.
3. Wierzchołki grafu G zostają rozdzielone na podzbiory $r_1 \cup r_2 \cup ... \cup r_m = V$, takie że $r_i \cap r_j = \emptyset$ dla każdego $i \neq j$. Do każdego zbioru należą tylko te wierzchołki, których lista poprzedników jest taka sama.
4. Każdemu zbiorowi r_i przyporządkowuje się pewną liczbę naturalną:
   i. jeśli istnieje łuk (v_a, v_b), taki że $v_b \in r_k$ oraz $v_a \in p_l$, to przyporządkuj zbiorowi r_k tą samą liczbę, którą przyporządkowano zbiorowi p_l.
   ii. jeśli żaden łuk nie spełnia powyższego warunku, to przyporządkuj zbiorowi r_i najniższy unikalny numer (najmniejsza liczba naturalna, która nie jest związana z żadnym zbiorem p_i ani ze zbiorem r_i).
5. Zgodnie z zasadą dwudzielności, wierzchołki grafu są podzielone na dwa zbiory $V = V_1 \cup V_2$.
6. Etykiety są słowami nad alfabetem $L_1 = a, b, c$. Etykiety przyporządkowuje się wszystkim wierzchołkom ze zbioru V w następujący sposób:
   i. każdemu wierzchołkowi $v_i \in V_1$ przyporządkowuje się etykietę $l(v_i) =, , b''$;
   ii. każdemu wierzchołkowi $v_i \in V_2$ przyporządkowuje się etykietę $l(v_i) =, , a''$;
   iii. Niech S będzie zbiorem słów nad alfabetem L_1 takich, że $S = c^*$. dla każdego wierzchołka $v_i \in V$, gdzie $v_i \in p_j$ wybierz słowo $s \in S$, takie że $length(s) = p_j$. Przyporządkuj $l(v_i) = L(v_i) + s$, gdzie „+” jest operacją konkatenacji;
   iv. dla każdego wierzchołka $v_i \in V$, gdzie $v_i \in r_j$, wybierz słowo $s \in S$, takie że $length(s) = r_j$. Przyporządkuj $l(v_i) = s + L(v_i)$, gdzie „+” jest operacją konkatenacji;

v. etykiecie każdego wierzchołka $v_i \in V_1$ przyporządkuj: $l(v_i) = (v_i)+, , a''$, gdzie „+" jest operacją konkatenacji;

vi. etykiecie każdego wierzchołka $v_i \in V_2$ przyporządkuj: $l(v_i) = (v_i)+, , b''$, gdzie „+" jest operacją konkatenacji;

Graf zaetykietowany według powyższej procedury może reprezentować wyniki rozważanego eksperymentu biochemicznego. Litera „a" reprezentuje aminokwas, po którym następuje cięcie w eksperymencie z wykorzystaniem pierwszej endopeptydazy a litera „b" reprezentuje aminokwas, po którym następuje cięcie w eksperymencie z wykorzystaniem drugiej endopeptydazy.

Do zaetykietowania grafu wykorzystano 3 litery otrzymując uproszczony graf peptydowy. Można pokazać, że graf zaetykietowany według zaproponowanej procedury za pomocą $k-$literowego alfabetu, można również zaetykietować za pomocą $k+1$ literowego alfabetu. W tym celu alfabet L_1 rozszerza się o dodatkową literę u i wykonuje się dowolny z poniższych kroków:

- Do zbioru liter symbolizujących aminokwasy, po których tnie endopeptydaza w pierwszym eksperymencie dodaje się „u", losowe wystąpienia litery „a" zostają zastąpione literą „u".

- Do zbioru liter symbolizujących aminokwasy, po których tnie endopeptydaza w drugim eksperymencie dodaje się „u", losowe wystąpienia litery „b" zostają zastąpione literą „u".

- Dla wszystkich wierzchołków z wybranego zbioru p_i sufiks etykiety c^*a zostaje zamieniony na ua. Analogiczną zmianę przeprowadza się dla prefiksów wszystkich wierzchołków ze zbioru r_j, który ma przyporządkowaną tą samą liczbę co zbiór p_i.

W ten sposób etykiety zbudowane nad alfabetem 3-literowym można rozszerzyć do etykiet zbudowanych nad alfabetem 20-literowym, uzyskując graf peptydowy. Aby otrzymany graf reprezentował rzeczywisty wynik eksperymentu biochemicznego, wymagane jest uwzględnienie jeszcze jednej uwagi.

W wyniku działania zaproponowanej metody etykietującej może się zdarzyć, że dwa wierzchołki będą miały taką samą etykietę. Sytuacja taka ma miejsce, gdy lista następników tych dwóch wierzchołków jest taka sama oraz lista poprzedników tych wierzchołków jest taka sama.

Poniżej pokazano jak można zmodyfikować metodę, aby uniknąć powtarzania się etykiet. Graf nie zawierający powtarzających się etykiet może reprezentować sytuację, gdy eksperyment chemiczny nie dostarcza informacji o powtórzeniach. Jeśli w wyniku procedury przyporządkowującej etykiety, istnieje zbiór wierzchołków o takiej samej etykiecie, to:

- Gdy wierzchołki należą do zbioru V_1, to literę b we wszystkich etykietach należy zastąpić przez słowo, którego pierwsza i ostatnia litera to „b" i słowo to nie zawiera żadnej innej litery „b", tak aby wybrane słowo było unikalne dla każdego z tych wierzchołków.

- Gdy wierzchołki należą do zbioru V_2, to literę a we wszystkich etykietach należy zastąpić przez słowo, którego pierwsza i ostatnia litera to „a" i słowo to nie zawiera żadnej innej litery „a", tak aby wybrane słowo było unikalne dla każdego z tych wierzchołków.

Podsumowując, dla dowolnej etykiety $\epsilon(v)$ można wyróżnić 3 podłańcuchy:

- prefiks, który determinuje łuki wchodzące do wierzchołka v,
- sufiks, który determinuje łuki wychodzące z wierzchołka v,
- część pomiędzy prefiksem a sufiksem, która może zapewnić unikalność wszystkich etykiet.

Pokazano, że każdy dwudzielny graf sprzężony jest grafem peptydowym [26][6]. Aby dla dowolnego grafu stwierdzić czy jest grafem peptydowym, należy sprawdzić czy jest spójnym grafem dwudzielnym i jednocześnie sprzężonym.

Sprawdzenie, czy dany graf jest grafem sprzężonym, można przeprowadzić w czasie wielomianowym. Dla każdej pary wierzchołków grafu przeprowadzane jest porównanie zbioru następników tych wierzchołków. Aby graf był grafem sprzężonym, to dla każdej pary wierzchołków zbiór następników musi być taki sam albo ich iloczyn musi być zbiorem pustym. Sprawdzenie to można przeprowadzić w czasie $O(n^3)$, gdzie n oznacza liczbę wierzchołków grafu. Algorytm 1 sprawdza, czy graf jest grafem sprzężonym, przy założeniu, że reprezentacją grafu jest macierz sąsiedztwa. Istnieniu łuku pomiędzy dwoma wierzchołkami odpowiada w macierzy wartość 1, w przeciwnym wypadku wartość w macierzy to 0. Po wykonaniu algorytmu zmienna „sprzezony" przechowuje iformację, czy graf jest grafem sprzężonym.

Algorithm 1 Algorytm sprawdzający czy graf jest grafem sprzężonym

```
1:  sprzezony = true;
2:  for i:=1 to n do
3:      for j:=i to n do
4:          wspolny = false;
5:          rozny = false;
6:          for k:=1 to n do do
7:              if M[i][k] == M[j][k] then
8:                  wspolny = true
9:              end if
10:             if M[i][k] != M[j][k] then
11:                 rozny = true;
12:             end if
13:         end for
14:         if wspolny AND rozny then
15:             sprzezony:=false;
16:         end if
17:     end for
18: end for
```

Sprawdzenie czy graf jest dwudzielny można sprowadzić do kolorowania wierzchołkowego grafu dwoma kolorami. Jeśli istnieje takie pokolorowanie grafu to jest on dwudzielny. Problem kolorowania grafu dwoma kolorami jest łatwy obliczeniowo. Algorytm 2 sprawdza możliwość wierzchołkowego pokolorowania grafu dwoma kolorami i jego spójność w czasie liniowym. Algorytm ten należy wykonać dla nieskierowanej wersji grafu. Po

zakończeniu działania algorytmu zmienna „spojny” oraz zmienna „kolorowalny” przechowują informację kolejno o tym czy graf jest spójny oraz czy daje się pokolorować wierzchołkowo dwoma kolorami.

Algorithm 2 Procedura sprawdzająca, czy graf jest spójny i można go pokolorować dwoma kolorami

```
procedure KOLORUJ(v, kolor)
    kolor(v) = kolor;
    if kolor = biały then
        kolor2 = czarny
    else
        kolor2 = biały
    end if
    for all nastepnik(v) do
        if kolor(nastepnik(v)) == kolor then
            kolorowalny := false;
        end if
        if kolor(nastepnik(v)) == brak then
            koloruj(nastepnik(v),kolor2)
        end if
    end for
end procedure
dwudzielny = true
spojny = true
koloruj(v_1,czarny)
for all v do
    if kolor(v) == brak then
        spojny = false;
    end if
end for
```

Podsumowując, sprawdzenie dla dowolnego grafu, czy jest grafem peptydowym jest problemem łatwym obliczeniowo. W pracy przedstawiono algorytm, który rozwiązuje to zagadnienie w czasie $O(n^3)$.

6.4 Przypadek bez błędów z dodatkową informacją o rozkładzie aminokwasów

Rozważany jest eksperyment biochemiczny bez błędów, gdzie dodatkowo przeprowadzono pełne trawienie w celu otrzymania informacji o liczności (rozkładzie) poszczególnych aminokwasów w oryginalnej cząsteczce. Zostanie zaproponowany wielomianowy algorytm dokładny rozwiązujący ten przypadek. Zostało pokazane, że przypadek bez błędów i bez informacji o rozkładzie aminokwasów jest łatwy obliczeniowo [26][6]. Analizę można więc sprowadzić do odpowiedzi na pytanie czy dodanie dodatkowej informacji ma wpływa na złożoność obliczeniową tak zmodyfikowanego problemu.

Rozwiązaniem w przypadku sytuacji idealnej jest dowolna skierowana ścieżka Hamiltona w grafie peptydowym. W przypadku, gdy dana jest informacja o liczności aminokwasów, rozwiązaniem problemu jest taka skierowana ścieżka Hamiltona w grafie, dla której powiązany z nią łańcuch znaków posiada oczekiwany rozkład liter.

Zostanie wprowadzony graf $G' = (V, A')$, który jest modyfikacją grafu peptydowego $G = (V, A)$. W G' wprowadza się łuki pomiędzy wierzchołkiem odpowiadającym ostatniej krótkiej sekwencji a każdym innym wierzchołkiem tego grafu. Nowe łuki mają długość 0. Skoro istnieje skierowana ścieżka Hamiltona w G, to musi istnieć skierowany cykl Hamiltona w G'.

Trzeba zauważyć, że suma wag łuków każdego skierowanego cyklu Hamiltona w G' jest taka sama. Wynika to z faktu, że niezależnie od położenia wierzchołka w permutacji, zawsze wychodzi i wchodzi do tego wierzchołka łuk o tej samej wadze. Usuwając jeden dowolny łuk ze skierowanego cyklu Hamiltona, otrzymuje się skierowaną ścieżkę Hamiltona. Suma wag tej skierowanej ścieżki jest równa sumie wag skierowanego cyklu, pomniejszonej o wagę usuniętego łuku.

Zostanie rozważony przypadek pesymistyczny, gdy graf G' jest dwudzielnym grafem pełnym - ma $\frac{n^2}{4}$ łuków. Skoro każdy skierowany cykl Hamiltona w G' ma tą samą sumę wag (długość), to może istnieć co najwyżej $\frac{n^2}{4}$ różnych długości skierowanych ścieżek Hamiltona w G', w zależności który łuk zostanie usunięty. Pomimo, że liczba skierowanych ścieżek Hamiltona w G' może być wykładnicza względem liczby wierzchołków, to liczba możliwych długości tych skierowanych ścieżek jest wielomianowo ograniczona względem liczby wierzchołków. Trzeba zauważyć, że spostrzeżenie to przekłada się na liczbę możliwych do uzyskania długich sekwencji związanych z tymi ścieżkami. Pomimo, że liczba uzyskanych skierowanych ścieżek Hamiltona może być wykładnicza względem liczby wierzchołków $|V|$, to istnieje co najwyżej $\frac{n^2}{4}$ możliwych do uzyskania sekwencji.

Usunięcie łuku ze skierowanego cyklu Hamiltona można sprowadzić do wskazania, który wierzchołek grafu będzie pierwszy, a który wierzchołek będzie ostatni w uzyskanej skierowanej ścieżce Hamiltona. Zakładając, że z cyklu usunięto łuk (v_i, v_j), uzyskana skierowana ścieżka Hamiltona rozpoczyna się wierzchołkiem v_j, a ostatnim wierzchołkiem tej ścieżki jest v_i. Podsumowując, wybór dwóch wierzchołków grafu jako wierzchołka rozpoczynającego i kończącego skierowaną ścieżkę Hamiltona w grafie jest równoznaczny z wyborem długości ścieżki Hamiltona w G' - każda skierowana ścieżka Hamiltona pomiędzy tymi dwoma wierzchołkami ma taką samą długość, a co za tym idzie, każda sekwencja uzyskana na podstawie tego rozwiązania ma taki sam rozkład aminokwasów.

Znalezienie skierowanej ścieżki Hamiltona w grafie G' jest problemem łatwym - sprowadza się do rozwiązania problemu asemblacji bez błędów, który został przedstawiony w poprzednim podrozdziale. Rozwiązanie rozważanego w tym podrozdziale problemu sprowadza się do znalezienia n^2 różnych skierowanych ścieżek Hamiltona w G' (pomiędzy każdą parą wierzchołków tego grafu) oraz wybraniu tej skierowanej ścieżki, dla której uzyskana sekwencja spełnia wymaganie dotyczące rozkładu aminokwasów. Problem asemblacji bez błędów z dodatkową informacją o rozkładzie aminokwasów jest więc łatwy obliczeniowo.

6.5 Przypadek bez wszystkich cięć

6.5.1 Definicja problemu i model matematyczny

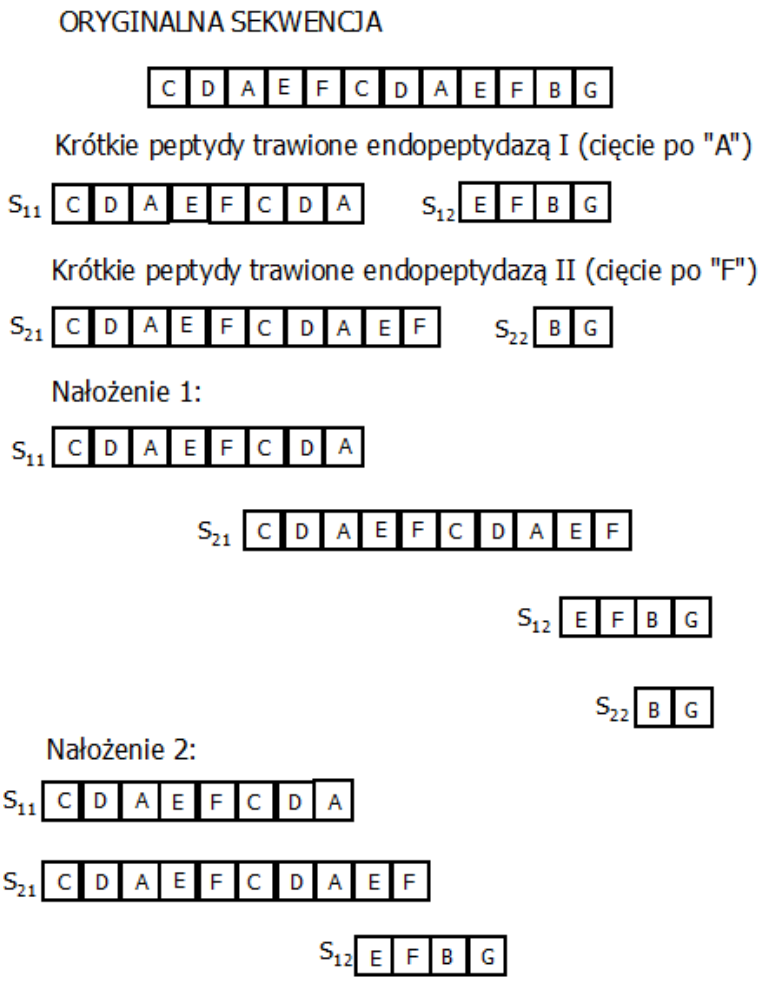
W przypadku braku wszystkich cięć sytuacja jest znacznie bardziej skomplikowana. Główna różnica polega na tym, że nie można zakładać maksymalnego nałożenia się

dwóch łańcuchów. Należy rozważyć każde możliwe nałożenie dwóch łańcuchów, które poprzedzone jest aminokwasem, po którym mogło nastąpić cięcie.

W przypadku idealnym, gdy zachodzą wszystkie cięcia i nie ma powtórzeń, to każde miejsce w oryginalnej cząsteczce jest pokryte krótkimi łańcuchami dokładnie dwukrotnie - raz w każdym zbiorze peptydów. Gdy nie ma wszystkich cięć to może wystąpić wielokrotne pokrycie oryginalnej cząsteczki krótkimi peptydami. Nie jest zatem możliwe odgórne stwierdzenie czy pewna sekwencja, która jest podsekwencją innej, rzeczywiście pokrywa ten sam fragment oryginalnej cząsteczki a co za tym idzie może być usunięta z dalszych rozważań.

Podsumowując, dwie zasadnicze zmiany w stosunku do sytuacji idealnej to:

- Mogą istnieć dowolne nałożenia dwóch sekwencji z innych zbiorów, które są poprzedzone aminokwasami, po których mogło zajść cięcie.
- Jeśli dana sekwencja zawiera się w pewnej sekwencji z innego zbioru, to nie jest to wystarczająca przesłanka, aby usunąć tą sekwencję.



Rysunek 6.3: Przykładowa sekwencja i jej asemblacja dla problemu bez wszystkich cięć (rysunek własny)

W rozprawie rozważa się problem, który możliwie najlepiej opisuje rzeczywistą sytuację. Zakłada się, że mogą istnieć błędy negatywne wynikające z powtórzeń. Dodatkowo

znany jest rozkład aminokwasów asemblowanej cząsteczki, który można uzyskać przeprowadzając pełne trawienie.

Podobnie, jak wcześniej, krótkim peptydom przyporządkowuje się odpowiadające im słowa zbudowane nad 20-literowym alfabetem Σ. Problem można zdefiniować następująco:

Problem 6.1.
Instancja: Multizbiór S ciągów znaków nad alfabetem Σ (Σ-zbiór wszystkich symboli reprezentujących poszczególne aminokwasy), rozkład D symboli z alfabetu Σ, np. zbiór par (x, i) dla wszystkich symboli x z alfabetu Σ, gdzie i jest liczbą całkowitą dodatnią.
Odpowiedź: Superciąg dla multizbioru ciągów S, spełniający rozkład D.

Rozważany problem jest silnie NP-trudny [27]. Nie istnieją wielomianowe algorytmy rozwiązujące ten problem, jeśli P $\neq$ NP.

Wyniki tej wersji eksperymentu biochemicznego można przedstawić za pomocą grafu $G'' = (V'', A'')$. Wierzchołki V'' tego grafu odpowiadają krótkim peptydom. Przyporządkowuje się im etykiety - słowa utworzone nad alfabetem Σ i reprezentujące te peptydy. Podobnie, jak w przypadku idealnym, zbiór V'' jest podzielony zgodnie z zasadą dwudzielności: $V'' = V_1'' \cup V_2''$ oraz $V_1'' \cap V_2'' = \emptyset$. Każdy ze zbiorów V_1'', V_2'' odpowiada wynikom trawienia długiego łańcucha inną endopeptydazą.

Istnieje łuk (v_i, v, j) w G'', gdzie $(v_i \in V_1 \wedge v_j \in V_2) \vee (v_i \in V_2 \wedge v_j \in V_1)$, gdy istnieje $p > 0$, takie że spełnione są następujące warunki:

- $\forall_{q \in \{1,2,\ldots,p\}}\ s_i(k-1+q) = s_j(q)$.
- $s_i(k-1) \in C_2 \wedge v_i \in V_1'') \vee (s_i(k-1) \in C_1 \wedge v_i \in V_2'')$

Wartość p, dla której spełnione są powyższe warunki, to długość poprawnego nałożenia się etykiet dwóch wierzchołków. Dla pary wierzchołków (v_i, v_j) może istnieć więcej niż jedna taka wartość, czyli więcej niż jedno możliwe nałożenie ich etykiet. Przedstawiony graf jest więc multigrafem.

Każdemu łukowi przyporządkowuje się wagę. Waga w_i każdego łuku $a_i \in A$ zdefiniowano jako zbiór par (x, n) dla wszystkich symboli $x_i \in \Sigma$, gdzie n oznacza liczbę wystąpień odpowiedniej litery w tym fragmencie etykiety, który bierze udział w nałożeniu. w_i nie reprezentuje więc pojedynczej wartości (długości nałożenia), jak to miało miejsce w przypadku grafu modelującego sytuację idealną, tylko reprezentuje rozkład wszystkich aminokwasów w tym nałożeniu. Podobnie definiuje się wagi $w(v)$ dla wszystkich wierzchołków $v \in V''$, jako zbiór par (x, n) dla wszystkich symboli $x_i \in \Sigma$, gdzie n to liczba wystąpień danej litery w etykiecie wierzchołka v. Na potrzeby dalszych rozważań przydatne będzie również zdefiniowanie długości łuku. Długość łuku to liczba liter prefiksu następnika (poprzednika), który determinuje istnienie tego łuku.

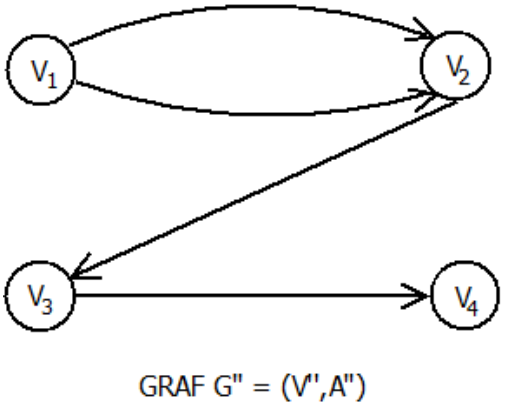
W sytuacji gdy dane są wszystkie cięcia, to każdy fragment asemblowanego łańcucha jest pokryty dokładnie dwa razy - raz w każdym zbiorze krótkich peptydów. W tym wypadku można usunąć peptydy, które zawierają się w sekwencjach z drugiego zbioru. Po usunięciu takiego peptydu, dany fragment łańcucha nadal jest jednokrotnie pokryty. W sytuacji, gdy nie ma wszystkich cięć, dany fragment łańcucha może być pokryty w spektrum wielokrotnie. Aby odtworzyć kolejność krótkich łańcuchów wystarcza, aby każde miejsce w głównym łańcuchu było pokryte co najwyżej dwukrotnie. Pozostałe fragmenty można usunąć.

Celem poniższych rozważań jest sprowadzenie problemu do sytuacji, gdzie każde miejsce w oryginalnej cząsteczce będzie pokryte co najwyżej przez dwie krótkie sekwencje. Dla każdego takiego wierzchołka dokonuje się więc decyzji, czy należy go usunąć czy

zachować. Jeśli graf G'' zawiera $|V| = n$ wierzchołków, to istnieje 2^n różnych podzbiorów wierzchołków, które można otrzymać w procesie decyzyjnym. Rodzina podgrafów indukowanych grafu G'' zawiera 2^n grafów. Musi istnieć co najmniej jeden graf α należący do tej rodziny, taki że peptydy odpowiadające wierzchołkom tego grafu pokrywają co najwyżej dwukrotnie dowolny fragment szukanej cząsteczki. Rozwiązanie problemu asemblacji zdefiniowane w teorii grafów polega na znalezieniu α oraz znalezienie takiej ścieżki Hamiltona w tym grafie, dla której związana z tą ścieżką sekwencja aminokwasów posiada szukany rozkład. Niech $\vec{D}$, $\vec{W}_i$ oraz $\vec{w}(v_i)$ będą 20-wymiarowymi wektorami związanymi z odpowiadającymi im parami (x, n). Aby przeprowadzić konwersję par do wektorów, zakłada się że pary zostały uporządkowane rosnąco alfabetycznie względem litery x. Każdy wymiar został oznaczony odpowiednią literą x. Wartość w tym wymiarze to odpowiadające n. Mając zdefiniowane wektory można wprowadzić wzór wyznaczający rozkład liter dla dowolnej ścieżki Hamiltona:

$$\vec{E} = \sum\nolimits_{\forall v \in \alpha} w(v) - \sum\nolimits_{\forall\, W_i \in R} W_i$$

gdzie R to zbiór łuków należących do ścieżki Hamiltona.



Rysunek 6.4: Graf G” dla przykładowej sekwencji z rysunku 6.3 (rysunek własny)

Opis grafu dla przykładu:

```
                        [A,B,C,D,E,F,G,...]
v1 -            w(v1) = [2,0,2,2,1,1,0,...]
etykieta = "CDAEFCDA"

v2 -            w(v2) = [2,0,2,2,2,2,0,...]
etykieta = "CDAEFCDAEF"

v3 -            w(v3) = [0,1,0,0,1,1,1,...]
etykieta = "EFBG"

v4 -            w(v4) = [0,1,0,0,0,0,1,...]
etykieta = "BG"
                w(a1) = [2,0,2,2,1,1,0,...]
                w(a2) = [1,0,1,1,0,0,0,...]
                w(a3) = [0,1,0,0,1,1,0,...]
                w(a4) = [0,1,0,0,0,0,1,...]
```

Warto przypomnieć twierdzenie „no free lunch". Zgodnie z nim, dla dowolnej funkcji celu nie da się określić, który algorytm optymalizacji jest najlepszy. Aby wybrać najlepszy algorytm optymalizacji dla konkretnej sytuacji, potrzeba dodatkowych założeń dotyczących występowania pewnych regularności w optymalizowanej funkcji. W praktyce wynika z tego twierdzenia, że warto zaimplementować kilka metod optymalizacyjnych, aby sprawdzić, która metoda najlepiej się sprawdzi w przypadku konkretnej funkcji celu.

W literaturze zaprezentowano dwie metody Tabu [10][7], które rozwiązują trudną wersją problemu asemblacji. Dla rozważanego w pracy NP-trudnego problemu asemblacji, przedstawiony zostanie algorytm ewolucyjny oraz metaheurystyka GRASP. Bazując na wynikach eksperymentu oraz właściwościach badanego problemu, w niniejszej pracy zaproponowano heurystykę dedykowaną.

6.5.2 Algorytm Ewolucyjny

Jak wspomniano, zaadaptowanie idei algorytmu ewolucyjnego do rozwiązania konkretnego problemu polega na zdefiniowaniu:

- sposobu kodowania
- środowiska (funkcji przystosowania)
- operatorów genetycznych
- schematu algorytmu

Sposób kodowania określa w jaki sposób zakodowane będą rozwiązania problemu, czyli osobniki oraz ich genotypy. Trzeba przypomnieć, że rozwiązaniem problemu jest skierowana ścieżka Hamiltona pewnym podgrafie indukowanym grafu G'', która spełnia równanie dotyczące rozkładu liter. Osobnikiem jest dowolna ścieżka w dowolnym grafie indukowanym grafu G''. Należy wprowadzić jednak pewne ułatwienie, aby znajdowanie rozwiązań dopuszczalnych (generowanie osobników) było problemem łatwym. Graf $H = (V'', A''')$ powstaje z grafu G'' w ten sposób, że między każdą parą wierzchołków, pomiędzy którymi nie ma żadnego łuku w G'', wprowadza się łuk o wektorze zerowym w H. Łuk ten reprezentuje brak nałożenia i położenie jednego krótkiego łańcucha za drugim. Dzięki tej modyfikacji graf H jest grafem pełnym. Znalezienie dowolnej skierowanej ścieżki Hamiltona w grafie pełnym jest problemem łatwym - jest to dowolna permutacja wierzchołków tego grafu. Podsumowując, rozwiązanie zostało zakodowane jako ciąg naprzemiennie występujących wierzchołków i łuków w dowolnym podgrafie indukowanym grafu H [7].

Funkcją przystosowania osobnika o_i do środowiska jest różnica w rozkładzie liter pomiędzy szukanym rozkładem a rozkładem liter w sekwencji związanej z tą ścieżką Hamiltona, która określa genotyp osobnika o_i:

$$f(o_i) = |\vec{D} - \vec{E}|$$

Funkcja ta jest minimalizowana.

W głównej pętli algorytmu ewolucyjnego wykonuje się kolejno reprodukcję, operatory genetyczne oraz sukcesję. Reprodukcja oraz sukcesja są wspólnie określane jako selekcja. Reprodukcja jest mechanizmem polegającym na doborze osobników do krzyżowania i mutacji w celu stworzenia nowej populacji. Osobniki o lepszym przystosowaniu są wybierane do procesu reprodukcji z większym prawdopodobieństwem. Do wyboru osobników

do procesu reprodukcji wykorzystano metodę ruletkową, prawdopodobieństwo wyboru osobnika jest odwrotnie proporcjonalne do wartości jego funkcji przystosowania:

$$p(o_i) = \frac{f_{\max} - f(o_i)}{\sum_m (f_{\max} - f(m))}$$

W procesie reprodukcji wybierane są dokładnie dwa osobniki. Sukcesja określa sposób utworzenia nowego pokolenia osobników. Jeśli, uzyskany w wyniku działania operatorów genetycznych, nowy osobnik jest lepiej przystosowany do środowiska niż osobnik najsłabiej przystosowany osobnik populacji, to nowy osobnik zastępuje tego osobnika i w ten sposób zostaje utworzone nowe pokolenie.

Operatory genetyczne to krzyżowanie oraz mutacja. Krzyżowanie pozwala na wymianę informacji pomiędzy osobnikami. W ramach eksperymentu obliczeniowego przetestowano dwa znane operatory krzyżowania: PX oraz LOX. Jednak najlepsze wyniki uzyskano dla operatora krzyżowania, którego działania przedstawia 3 [7].

Algorithm 3 Operator krzyżowania

1: Wybierz 2 osobniki X_i oraz X_j do krzyżowania według zasad selekcji ruletkowej
2: S := wszystkie wspólne podciągi X_i i X_j
3: NewSolution = pobierz losowo element z S
4: **for** i:=2 to rozmiar(S) **do**
5: NewSolution = NULL;
6: Pobierz losowo element z S
7: Wstaw go na takiej losowej pozycji j w NewSolution, że element NewSolution(j-1) zawiera niezerowe nałożenie z elementem NewSolution(j) oraz element NewSolution(j) zawiera niezerowe nałożenie z elementem NewSolution(j+1)
8: Jeśli było to niemożliwe, to wstaw go na takiej losowej pozycji NewSolution(j), że istnieje niezerowe nałożenie z NewSolution(j-1) lub NewSolution(j+1)
9: Jeśli to niemożliwe to wstaw go na losowej pozycji w NewSolution
10: **end for**

Jako niezerowe nałożenie rozumie się łuk o niezerowej wadze pomiędzy ostatnim wierzchołkiem jednej podścieżki a pierwszym wierzchołkiem drugiej podścieżki. Dla dwóch wybranych do krzyżowania osobników X_i oraz X_j tworzony jest zbiór S zawierający wszystki wspólne podścieżki dla tych rozwiązań. Poniższy przykład prezentuje sposób budowania S:

$$X_i = v_1\, e_1\, v_2\, e_2\, v_3\, e_3\, v_5\, e_4\, v_4$$

$$X_j = v_1\, e_1\, v_2\, e_2\, v_3\, e_3\, v_4\, e_5\, v_5$$

Zbiór $S = \{v_1\, e_1\, v_2\, e_2\, v_3, v_4, v_5\}$

Nowe rozwiązanie jest budowane w oparciu o zbiór wspólnych podścieżek S. Dopóki S nie jest pusty, losowo wybierana jest podścieżka, którą usuwa się ze zbioru i umieszcza w tymczasowym rozwiązaniu NewSolution. NewSolution to ciąg wspólnych podścieżek połączonych łukami. Dodając nową podścieżkę do rozwiązania rozważa się jej dodanie w dowolnym miejscu tego ciągu (na początku, na końcu lub pomiędzy dwoma innymi podścieżkami w środku ciągu). Dla każdego miejsca w podciągu, gdzie może zostać umieszczona podścieżka, należy rozważyć wszystkie możliwe łuki pomiędzy wierzchołkami tej

podścieżki a wierzchołkami ścieżek sąsiednich a następnie:

- Jeśli istnieją takie położenia podścieżki w ciągu, że do pierwszego wierzchołka podścieżki wchodzi niezerowy łuk ORAZ z ostatniego wierzchołka podścieżki wychodzi niezerowy łuk, to jako poprawne nałożenie wybierane jest z równym prawdopodobieństwem jedno z takich nałożeń wraz z tymi łukami; w przeciwnym wypadku
- Jeśli istnieją takie położenia podścieżki w ciągu, że do pierwszego wierzchołka podścieżki wchodzi niezerowy łuk ALBO z ostatniego wierzchołka podścieżki wychodzi niezerowy łuk, to jako poprawne nałożenie wybierane jest z równym prawdopodobieństwem jedno z takich nałożeń wraz z tymi łukami; w przeciwnym wypadku:
- Wybierane jest z równym prawdopodobieństwem dowolne nałożenie wraz z łukami zerowymi

Zaproponowany probabilistyczny operator krzyżowania daje znacznie lepsze wyniki niż dwa popularne operatory krzyżowania.

Mutacja to wymiana fragmentów kodu w jednym osobniku. Prawdopodobieństwo mutacji zostało określone eksperymentalnie i wynosi $p = 0.05$. Pseudokod 4 pokazuje działanie mutacji dla pewnego osobnika X_k.

Algorithm 4 Pseudokod mutacji

1: **procedure** MUTACJA(X_k)
2: Wybranie losowo 2 wierzchołków v_i oraz v_j w rozwiązaniu X_k
3: zamiana wierzchołków miejscami
4: Wybranie losowo (jeśli istnieją) niezerowych łuków pomiędzy wstawionymi wierzchołkami a ich sąsiadami
5: Wybranie łuków o wartości "0"między tymi wstawionymi wierzchołkami a ich sąsiadami, pomiędzy którymi nie istnieją łuki niezerowe
6: **end procedure**

Zaproponowany operator krzyżowania, który preferuje rozwiązania o niezerowych łukach, uzyskał najlepsze wyniki podczas eksperymentu obliczeniowego. Można to wytłumaczyć tym, że korzysta on z dodatkowej wiedzy dotyczącej problemu - kiedy to możliwe wybiera niezerowe łuki. Tylko takie łuki mogą modelować rzeczywiste częściowe nałożenie dwóch krótkich peptydów.

Należy zwrócić uwagę, że preferowanie łuków o wyższych wartościach powinno prowadzić do lepszego rozwiązania. Istnienie łuku pomiędzy dwoma wierzchołkami modeluje jedną z dwóch poniższych sytuacji:

- Odpowiadające im peptydy rzeczywiście pokrywają ten sam fragment cząsteczki, a co za tym idzie częściowo się pokrywają nawzajem,
- Odpowiadające im peptydy pokrywają całkowicie inne fragmenty oryginalnej cząsteczki, jednak prefiks etykiety drugiego wierzchołka jest równy sufiksowi etykiety pierwszego wierzchołka.

Druga sytuacja oznacza, że w rzeczywistym eksperymencie drugi krótki peptyd nie występuje bezpośrednio za pierwszym, jednak istnieje między nimi łuk, który modeluje takie niepoprawne biologicznie nałożenie się dwóch sekwencji. Prawdopodobieństwo, że wystąpi łuk o długości k, który nie opisuje rzeczywistego nałożenia się dwóch sekwencji wynosi:

$$p(|w(v_i, v_j)| = k) = (\frac{1}{20})^k$$

Jest to prawdopodobieństwo, że pierwsze k liter etykiety następnika jest takie samo jak l ostatnich liter poprzednika. Przy założeniu, że nałożenie tych etykiet nie modeluje rzeczywistej sytuacji biologicznej, tzn. krótkie peptydy nie pokrywają tego samego fragmentu cząsteczki. Z powyższego wzoru wynika przykładowo, że istnienie łuku o długości 2, który nie modeluje rzeczywistego nałożenia się dwóch peptydów wynosi $p = 0.0025$, podczas gdy istnienie takiego łuku o długości 4 określa prawdopodobieństwo $p = 0.00000625$. Podsumowując - prawdopodobieństwo występowania łuków, które nie modelują rzeczywistych nałożeń dwóch peptydów maleje wykładniczo z długością łuku.

Obserwacja ta została wykorzystana podczas projektowania algorytmu ewolucyjnego. Trzeba przypomnieć, że dla wszystkich wierzchołków należy rozważyć ich usunięcie z grafu. Prawdopodobieństwo, że pewne dwie sekwencje $s1$ oraz $s2$, gdzie $s1$ jest podsekwencją $s2$, nie pokrywają tego samego fragmentu oryginalnego peptydu można wyrazić takim samym wzorem jak powyżej, gdzie k jest długością etykiety wierzchołka związanego z $s1$. Można zatem zauważyć, że prawdopodobieństwo to jest bardzo niskie.

Rozwiązania poszukuje się w pewnym podgrafie indukowanym H' grafu H. Dla każdego wierzchołka można rozważyć jego usunięcie z H. Wierzchołkom grafu H nadaje się kolory. Podejście to zostało to również przedstawione w pracy [29]:

- każdemu wierzchołkowi, którego etykieta zawiera się w etykiecie innego wierzchołka przyporządkuj kolor czarny,
- wierzchołkom, którym nie przydzielono dotychczas koloru, przyporządkuj kolor biały.

Podgraf indukowany H' grafu H, który zawiera wszystkie białe i tyle te wierzchołki jest grafem wejściowym do działania algorytmu ewolucyjnego, którego działanie przedstawia pseudokod 5. Z wierzchołkami w kolorze czarnym związane są te sekwencje, które z dużym prawdopodobieństwem zawierają się w innych, dłuższych sekwencjach. Sekwencje takie można więc usunąć z dalszych rozważań.

Zmienna „rozwiązanie" zawiera osobnika o najlepszej wartości funkcji przystosowania. P_n to n-te pokolenie (populacja) osobników. T_n to populacja tymczasowa na której przeprowadza się działania operacji genetycznych. Zmienna „liczba_przegladanych_grafow" oznacza liczbę grafów rozpinających grafu H, bazując na których zostaną utworzone populacje natomiast zmienna liczba_iteracji określa liczbę generacji (pokoleń populacji), które zostaną utworzone dla każdego grafu indukowanego. Wartości tych dwóch zmiennych zostały dobrane w eksperymencie obliczeniowym.

6.5.3 Algorytm GRASP

Pokazano, że preferowanie długich łuków w grafie H może prowadzić do znajdowania lepszych rozwiązań. Prowadzi to do wniosku, że wybranie jako punkt startowy dla metody optymalizacyjnej rozwiązania zawierającego możliwie długie łuki i dalsza optymalizacja takiego rozwiązania powinno prowadzić do znalezienia dobrego rozwiązania. Obserwacje te doprowadziły do wyboru metaheurystyki GRASP. Główną ideą GRASP jest stworzenie dobrego rozwiązania początkowego, a następnie jego lokalna optymalizacja. Eksperyment obliczeniowy przeprowadzony dla algorytmu ewolucyjnego pokazał, że w ponad 90% przypadków najlepsze rozwiązanie znajdowano w tym podgrafie indukowanym H', który zawiera wszystkie białe wierzchołki H i nie zawiera żadnego czarnego wierzchołka.

Algorithm 5 Algorytm ewolucyjny

```
1: H' = podgraf indukowany grafu H zawierający tylko białe wierzchołki
2: rozwiazanie = dowolny losowy osobnik;
3: for m=1 to liczba_przegladanych_grafow do
4:     inicjacja P_0                          ▷ zainicjuj populację w oparciu o graf H
5:     for n=1 to liczba_iteracji do
6:         New = Krzyżowanie(P_n)
7:         T_n = P_n
8:         if f(New) < f(X_i) OR f(New) ¡f(X_j) then
9:             if f(X_i) < f(X_j) then
10:                T_n = zastąp w populacji P_n osobnika X_j przez New
11:            else
12:                T_n = zastąp w populacji P_n osobnika X_i przez New
13:            end if
14:        end if
15:        T_n = Mutacja w populacji T_n (zachodzi z prawdopodobieństwem 5%)
16:        P_{n+1} = T_n
17:    end for
18:    if Najleszpy osobnik P_n jest lepszy od rozwiazanie then
19:        rozwiązanie = najlepszy osobnik P_n
20:    end if
21:    H' = podgraf indukowany grafu H, z którego usunięto losowo jeden wierzchołek
22: end for
```

Zdecydowano, że takie wyniki w przypadku projektowania metaheurystyki pozwalają ograniczyć rozważania tylko do tego grafu i w ten sposób skrócić czas obliczeń.

Niech zbiór U będzie zbiorem wszystkich białych wierzchołków grafu H. W każdym kroku algorytmu wybierany jest wierzchołek z U, który jest dodawany do $NewSolution$ i usuwany z U. Możliwości wyboru wierzchołka z U są ograniczone. Funkcją oceny heurystycznej f' jest funkcja obliczająca długość bieżącej ścieżki rozwiązania $NewSolution$. Dla wszystkich wierzchołków z U na każdym kroku sprawdzane są możliwe sposoby (możliwe łuki) dołączenia ich na początku lub na końcu $NewSolution$. W każdym kroku działania algorytmu tworzy się listę kandydatów RLC (Restricted Candidate List), która zawiera tylko te elementy (para wierzchołek + łuk), które powodują najwyższy przyrost wartości funkcji celu f'. Prawdopodobieństwo wyboru dowolnego elementu z RLC jest takie same. RLC zawiera wybory nie gorsze o więcej niż 35% od optymalnego wyboru w danym kroku algorytmu, jednak nie więcej niż 30% wszystkich par wierzchołek+łuk. Wartości tych dwóch parametrów zostały dobrane w eksperymencie obliczeniowym.

Algorytm 6 prezentuje tworzenie rozwiązania początkowego [46].

Uzyskane rozwiązanie początkowe NewSolution jest następnie optymalizowane. Jako sąsiedztwo rozwiązania zdefiniowano te rozwiązania, które mogą powstać po zamianie miejscami dwóch wierzchołków i wybraniu dowolnych łuków między tymi wierzchołkami a ich sąsiadami. Rozwiązania ocenia się za pomocą funkcji celu f obliczającą różnicę

Algorithm 6 I etap metody GRASP: tworzenie rozwiązania początkowego

```
U = wszystkie białe wierzchołki grafu H
NewSolution = NULL
while U zawiera wierzchołki do
    Utworzenie listy RLC
    Losowe wybranie elementu (pary wierzchołek + łuk) z RLC
    Usunięcie wybranego wierzchołka z U
    Dodanie wybranego elementu do NewSolution
end while
```

pomiędzy licznością liter w znalezionym rozwiązaniu a szukanym rozkładem. W procesie optymalizacji losowo wybierane jest rozwiązanie sąsiednie o niższej wartości funkcji celu. Prawdopodobieństwo wyboru danego rozwiązania jest wprost proporcjonalne do wartości o jaką zostanie polepszona wartość funkcji celu f. Dla danego rozwiązania X_i i zbioru jego sąsiadów X prawdopodobieństwo przejścia do rozwiązania $X_j \in X$, gdzie $f(X_j) < f(X_i)$, jest wyrażone wzorem:

$$P(X_j) = \frac{f(X_i) - f\,X_j)}{\sum_{\forall\, X_k \in X; f(X_k) < f(X_i)} (f(X_i) - g(X_k))}$$

Optymalizacja jest wykonywana tak długo, dopóki dla aktualnego rozwiązania zbiór jego sąsiadów o lepszej wartości funkcji f jest niepusty.

6.5.4 Heurystyka dedykowana

Pokazano, że istnienie łuków, które nie odzwierciedlają prawdziwego nałożenia się dwóch sekwencji, jest mało prawdopodobne. Prawdopodobieństwo to maleje wykładniczo wraz ze wzrostem długości łuku l. Prowadzi to do obserwacji, że graf G'' jest dość rzadki - posiada $|V| - 1$ łuków odzwierciedlających rzeczywiste nakładanie się krótkich sekwencji a istnienie innych łuków jest możliwe, lecz mało prawdopodobne. Można powiedzieć, że dla losowych danych liczba łuków jest rzędu $|V|$. Uwaga ta jest prawdziwa dla rzeczywistych sekwencji, gdzie rozkład aminokwasów jest w pełni losowy.

Zaproponowana w tym rozdziale heurystyka dedykowana bazuje na powyższej obserwacji a co za tym idzie jest przeznaczona głównie do asemblacji rzeczywistych sekwencji peptydowych.

Przedstawione uprzednio rozwiązania problemu asemblacji opierają się na przeszukiwaniu przestrzeni rozwiązań, która zawiera podgrafy indukowane grafu G''. W każdym takim grafie rozwiązaniem dopuszczalnym jest dowolna permutacja wierzchołków.

Uogólniając, w przypadku metaheurystyk rozwiązanie polega na znalezieniu w grafie takiej ścieżki w H, z którą związana jest sekwencja spełniająca warunek dotyczący rozkładu aminokwasów. Dwa sąsiednie peptydy w rozwiązaniu zawsze powinny się częściowo nakładać. Należy więc zauważyć, że znalezione rozwiązania, które zawierają łuki o wadze zero, z pewnością nie odzwierciedlają poprawnego nałożenia dwóch sąsiadujących ze sobą peptydów.

Zostanie zaproponowane podejście alternatywne. Przedmiotem rozważań będzie rodzina wszystkich podgrafów indukowanych grafu $R_1(G'')$. Dla każdego wierzchołka dowolnego grafu $I \in R_1(G'')$ podjęte zostają dwie decyzje: zostaje wybrana tylko jedna długość łuków wchodzących i wychodzących do tego wierzchołka. Otrzymany graf będzie

grafem sprzężonym, gdyż podobnie jak w przypadku idealnym, dla każdej etykiety określona zostanie określona jedna długość prefiksu i jedna długość sufiksu determinująca łuk.

Jeśli dla każdego wierzchołka pewnego grafu $I \in R_1(G'')$ decyzja dotycząca długości łuków wchodzących oraz wychodzących będzie podejmowana tylko spośród długości łuków, które rzeczywiście wchodzą i wychodzą z tego wierzchołka, to podjęcie tych decyzji dla wszystkich wierzchołków grafu, spowoduje powstanie grafu sprzężonego, który dodatkowo jest podgrafem rozpinającym grafu I. Rodzina wszystkich tych podgrafów rozpinających grafu I, które można otrzymać w ten sposób zostanie oznaczona $R_2(I)$.

W ten sposób można skonstruować podgraf rozpinający J podgrafu indukowanego I grafu G''. J zawiera wszystkie te łuki, które opisują rzeczywiste częściowe nakładanie się sekwencji. Znalezienie skierowanej ścieżki Hamiltona w grafie J jest rozwiązaniem problemu asemblacji. Należy przypomnieć, że znalezienie ścieżki Hamiltona w grafie sprzężonym jest problemem łatwym obliczeniowo.

Niech $R_1(G'')$ będzie rodziną wszystkich indukowanych podgrafów pewnego grafu G'' a $R_2(I)$ rodzina wszystkich sprzężonych rozpinających grafów pewnego grafu $I \in R_1(G'')$. Sposób działania metaheurystyki przedstawia algorytm .

Algorithm 7 Heurystyka dedykowana

```
1: for all I ∈ R1(G'') do
2:     for all J ∈ R2(I) do
3:         Wygenerowanie grafu O, dla którego J jest grafem sprzężonym
4:         if O zawiera drogę Eulera then
5:             Wygenerowanie sekwencji aminokwasowej s związanej z grafem J
6:             if s spełnia zadany rozkład aminokwasów then
7:                 Przyjęcie s jako rozwiązania problemu
8:                 Zakończenie działania algorytmu
9:             else
10:                Próba przywrócenia własności grafu eulerowskiego dla grafu O
11:                if O zawiera drogę Eulera then
12:                    Wygenerowanie sekwencji aminokwasowej s związanej z grafem J
13:                    if s spełnia zadany rozkład aminokwasów then
14:                        Przyjęcie s jako rozwiązania problemu
15:                        Zakończenie działania algorytmu
16:                    end if
17:                end if
18:            end if
19:        end if
20:    end for
21: end for
```

Komentarza wymaga sytuacja, gdy graf O nie jest grafem eulerowskim. Następuje wtedy próba przywrócenia własności grafu eulerowskiego dla grafu O.

!!!

graf 0 - oryginalny J - sprzezony !!!

Przypadkiem pesymistycznym dla powyższego algorytmu jest sytuacja, gdy graf G'' jest grafem pełnym i każdy łuk wchodzący i wychodzący z wierzchołka ma inną długość. Jeśli $|V| = n$ to dla każdego wierzchołka możliwych jest n różnych decyzji dotyczących długości łuku wchodzącego i wychodzącego. W tej sytuacji istnieje (n^2) możliwości dla jednego wierzchołka i $(n^2)^n$ grafów J.

Biorąc pod uwagę, że grafy przedstawiające wyniki eksperymentu są bardzo rzadkie, liczba sprzężonych podgrafów rozpinających grafu I powinna być niewielka. W ogólności wynosi ona: $P = |in_1| \cdot |in_2| \cdot ... \cdot |in_m| \cdot |out_1| \cdot |out_2| \cdot ... \cdot |out_m|$, gdzie in_i i out_i jest liczbą odpowiednio łuków wchodzących i wychodzących z wierzchołka v_i a m to liczba wierzchołków tego grafu.

Podsumowując, we wszystkich przedstawionych podejściach należy rozpatrzyć wszystkie grafy indukowane grafu G''. W przypadku przedstawionych metaheurystyk należy przejrzeć wszystkie permutacje wierzchołków każdego takiego grafu, natomiast w heurystyce dedykowanej wszystkie sprzężone podgrafy rozpinające każdego takiego grafu. Heurystyka dedykowana opiera się na założeniu, że w przypadku grafów opisujące rzeczywiste sekwencje aminokwasów $P << m!$.

Dodatkowo, punktem startowym heurystyki jest graf indukowany, który zawiera wszystkie wierzchołki białe i żadnego czarnego, gdyż pokazano, że prawdopodobieństwo że pewna etykieta zawiera się w innej etykiecie i nie odzwierciedla to prawdziwego nakładania się tych dwóch fragmentów sekwencji jest bardzo małe.

6.6 Klasyfikacja problemów asemblacji

W rozdziale tym zestawiono problemy asemblacji. Na potrzeby definicji problemów asemblacji zostanie zdefiniowane widmo idealne oraz widmo pełne [35].

W przypadku, gdy zachodzą wszystkie cięcia, które wynikają z działania endopeptydazy, powstałe spektrum nosi nazwę spektrum idealnego i jest oznaczone P^i.

Należy rozpatrzyć sytuację, gdy nie wszystkie cięcia zachodzą. Załóżmy, że cząsteczka posiada k aminokwasów, po których może nastąpić cięcie. Podejmując dla każdego z aminokwasów decyzję, czy nastąpi po nim cięcie czy nie, otrzymuje się pewien multizbiór krótkich fragmentów tego łańcucha. Należy zauważyć, że istnieje 2^k różnych procesów decyzyjnych prowadzących do różnego pocięcia sekwencji i powstania innych krótkich sekwencji. Załóżmy dodatkowo, że na każdej cząsteczce w roztworze przeprowadzono inny proces decyzyjny (uzyskano inne fragmenty z jej pocięcia) a w roztworze znajduje się na tyle dużo cząsteczek, aby przeprowadzić wszystkie możliwe procesy decyzyjne (co najmniej 2^k cząsteczek). Wynikiem takiego trawienia jest uzyskanie *pełnego widma*, czyli widma które zawiera wszystkie możliwe do uzyskania krótkie sekwencje. Pełne widmo oznaczone zostanie P^p.

Ze spektrum P^i zostanie powiązany odpowiadający mu zbiór słów S^i, natomiast ze P^p powiązany jest zbiór słów S^p.

Problem 6.2 opisuje przypadek idealny, gdy zachodzą wszystkie cięcia. Pokazano, że w tym wypadku rozwiązaniem jest ścieżka Hamiltona w grafie sprzężonym [6][26]. Problem tej jest łatwy obliczeniowo.

Problem 6.2.
Instancja: Multizbiór słów S_i nad alfabetem Σ **Odpowiedź:**
Superciąg dla zbioru słów S_i

Przypadek opisany w rozprawie, gdy nie zachodzą wszystkie cięcia i dany jest rozkład aminokwasów (problem 6.1) należy do klasy problemów silnie NP-trudnych [27]. W rozprawie zaproponowano trzy metody przybliżone do jego rozwiązania.

Problem 6.3 jest zwiazany z sytuacją, gdy nie zachodzą wszystkie cięcia, ale otrzymane spektrum jest nadzbiorem spektrum idealnego. Problem ten jest łątwy obliczeniowo [27]. Aby znaleźć rozwiązanie tego problemu w czasie wielomianowym, należy wskazać w zbiorze słów wszystkie te słowa, które odpowiadają fragmentom pochodzącym z pełnego trawienia a następnie wykorzystać algorytm zaproponowany dla przypadku idealnego. Słowa odpowiadające fragmentom pochodzącym z procesu pełnego trawienia można rozpoznać po tym, że zawierają co najwyżej jedną literę odpowiadającą aminokwasowi po którym następuje cięcie. Jest to ostatnia litera w tych słowach.

Problem 6.3.
Instancja: Multizbiór słów S nad alfabetem Σ, taki że $S \subseteq S_p$ oraz $S_i \subseteq S$. **Odpowiedź:**
Superciąg dla zbioru słów S.

Problem 6.4 opisuje sytuację, gdy zachodzą wszystkie cięcia i dodatkowo dany jest rozkład aminokwasów. W rozprawie pokazano, że dodanie rozkładu aminokwasu nie powoduje zmiany klasy złożoności. Problem jest łatwy obliczeniowo.

Problem 6.4.
Instancja: Multizbiór słów S_i nad alfabetem Σ; rozkład D symboli z alfabetu Σ tj. zbiór par (x, i) dla wszystkich symboli x z alfabetu Σ, gdzie i jest liczbą całkowitą dodatnią. **Odpowiedź:**
Superciąg dla zbioru słów S_i spełniający rozkład D

Problem 1 to sytuacja bez wszystkich cięć, gdy nie ma dodatkowej informacji dotyczącej rozkładu aminokwasów. Problem należy do klasy problemów NP-trudnych [27].

Problem 1.
Instancja: Multizbiór słów $S \in S_p$ nad alfabetem Σ. **Odpowiedź:**
Superciąg dla zbioru słów S.

6.7 Wnioski

Z punktu widzenia rozważań matematycznych, celem badań w dziedzinie asemblacji było zdefiniowanie modeli grafowych dla przypadku idealnego oraz dla przypadku z błędami mogącymi wystąpić w eksperymencie biochemicznym. Formalne zdefiniowanie klasy grafów peptydowych, które mogą reprezentować wyniki eksperymentu biochemicznego, pozwala na skupieniu się na badaniu własności grafów i oddzieleniu tych rozważań od analizy własności problemu biologicznego. Pokazanie izomorfizmu klasy grafów peptydowych z klasą dwudzielnych grafów sprzężonych pozwala na zastosowanie wszystkich twierdzeń dotyczących tych drugich przy badaniu własności grafów peptydowych. Jest to o tyle ważne, że grafy sprzężone były analizowane w literaturze nie tylko w kontekście modelowania wyników eksperymentów biologicznych.

W pracy udowodniono, że problem gdy powstają wszystkie cięcia i znany jest rozkład aminokwasów jest łatwy obliczeniowo. Zestawienie problemów asemblacji pozwala wyciągnąć wniosek, że dodatkowa informacja o rozkładzie aminokwasów w szukanej cząsteczce nie powoduje zmiany klasy obliczeniowej takiego problemu w porównaniu do odpowiadającej mu wersji bez tej dodatkowej informacji. Jest to cenna informacja dla biologów, że dodanie rozkładu nie powoduje istotnego wydłużenia czasu obliczeń, może natomiast stanowić dodatkową metodę sprawdzenia poprawności otrzymanych rozwiązań. Informację o liczności aminokwasów można łatwo uzyskać poprzez pełne trawienie peptydu.

Analizując wersję NP-trudną problemu asemblacji bez wszystkich cięć i ze znanym rozkładem przebadano możliwość wykorzystania różnych metaheurystyk. Zaimplementowano i przetestowano algorytm ewolucyjny oraz metodę GRASP. Wyniki porównano z wynikami metody Tabu w literaturze. Dzięki implementacji kilku metod można było wyciągnąć wnioski, że algorytm ewolucyjny w przypadku rozważanego problemu sprawdza się najlepiej.

Praca nad metodami metaheurystycznymi doprowadziła do dwóch innych istotnych wniosków: rozważane grafy peptydowe w przypadku asemblacji z błędami są relatywnie rzadkie, dodatkowo preferowanie w rozwiązaniu łuków o najwyższych wagach często prowadzi do znalezienia dobrej jakości rozwiązań. Te dwa wnioski pozwoliły na zaprojektowanie dedykowanej heurystyki, która pozwala na uzyskanie znacznie lepszych wyników.

Podsumowując, otrzymane wyniki stanowią nie tylko ciekawe przemyślenia z punktu widzenia rozważań teoretycznych, ale dzięki zaprojektowanym narzędziom i przedstawionym obserwacjom mogą przyczynić się do rozpowszechnienia zaproponowanego podejścia do ustalenia pierwszorzędowej budowy peptydów.

Rozdział 7

Wyniki eksperymentów obliczeniowych

7.1 Wstęp

Pokazano, że problem asemblacji bez wszystkich cięć i z dodatkową informacją o rozkładzie aminokwasów jest trudny obliczeniowo. Oznacza to, że nie istnieją wielomianowe algorytmy rozwiązujące ten problem jeżeli $P \neq NP$. Można jednak zaproponować metody, które znajdują przybliżone rozwiązanie w rozsądnym czasie. W praktyce wyniki metod przybliżonych są często wystarczające.

Celem eksperymentu obliczeniowego, którego wyniki przedstawione są w niniejszym rozdziale było sprawdzenie skuteczności zaproponowanych metod. Aby umożliwić ich porównanie, testy wszystkich algorytmów zostały przeprowadzone na tych samych instancjach i w tym samym środowisku. Stworzono te same warunki co dla metody Tabu opisanej w literaturze [7]. Umożliwi to porównanie nowych algorytmów ze wspomnianą metodą Tabu.

7.2 Parametry algorytmów

Parametry wszystkich algorytmów dobrano eksperymentalnie w pierwszej fazie eksperymentu obliczeniowego.

Dla algorytmu ewolucyjnego:

- wielkość populacji 10000,
- liczba iteracji 10000,
- liczba przeszukiwanych grafów 30,
- prawdopodobieństwo mutacji 0.05.

Dla metody GRASP:

- RLC zawiera rozwiązania nie gorsze o więcej niż 35% od rozwiązania optymalnego,
- RLC zawiera nie więcej niż 30 % wszystkich rozwiązań.

7.3 Porównanie metaheurystyk

Wszystkie algorytmy zostały zaimplementowane w języku Java 1.5 i przetestowane na komputerze klasy PC z procesorem Intel 2xXenon 3.6 GHz z 4 GB RAM. Testy zostały wykonane w środowisku Linux. Na potrzeby eksperymentu utworzono 3 zbiory sekwencji.

Pierwszy zbiór zawiera peptydy o losowej sekwencji aminokwasów, gdzie każdy aminokwas ma równe prawdopodobieństwo ($p = 0.05$) wystąpienia. Zbiór 450 sekwencji został podzielony na 45 podzbiorów, w zależności od długości sekwencji, liczby aminokwasów (miejsc w sekwencji), po których powinny wystąpić cięcia oraz procentu błędnych sytuacji, gdy cięcia nie zaszły. W każdym podzbiorze umieszczono 10 sekwencji. Tabela 7.1 przedstawia podział pierwszego zbioru peptydów na podzbiory:

Tabela 7.1: Podział pierwszego zbioru peptydów na podzbiory.

Lp.	Długość sekwencji	Miejsca cięć	Liczba błędów
1	100	10	10%
2	100	10	20%
3	100	10	30%
4	100	15	10%
5	100	15	20%
6	100	15	30%
7	100	20	10%
8	100	20	20%
9	100	20	30%
10	150	10	10%
11	150	10	20%
12	150	10	30%
13	150	15	10%
14	150	15	20%
15	150	15	30%
16	150	20	10%
17	150	20	20%
18	150	20	30%
19	200	10	10%
20	200	10	20%
21	200	10	30%
22	200	15	10%
23	200	15	20%
24	200	15	30%
25	200	20	10%
26	200	20	20%
27	200	20	30%
28	250	10	10%
29	250	10	20%
30	250	10	30%
31	250	15	10%
32	250	15	20%
33	250	15	30%
34	250	20	10%

Lp.	Długość sekwencji	Miejsca cięć	Liczba błędów
35	250	20	20%
36	250	20	30%
37	300	10	10%
38	300	10	20%
39	300	10	30%
40	300	15	10%
41	300	15	20%
42	300	15	30%
43	300	20	10%
44	300	20	20%
45	300	20	30%

Drugi zbiór również zawiera losowe sekwencje aminokwasów. Różnica w sposobie przygotowania danych polega na tym, że w pierwszym zbiorze lista miejsc, w których zachodzą cięcia, została z góry określona natomiast w drugim wykorzystano dwie sztuczne endopeptydazy: jedna rozpoznaje kwas asparaginowy, a drugą prolinę. Drugi zbiór zawiera 150 sekwencji, które zostały podzielone na podzbiory zawierające po 10 sekwencji, zgodnie z tabelą 7.2.

Tabela 7.2: Podział drugiego zbioru peptydów na podzbiory.

Lp.	Długość sekwencji	Liczba błędów
1	100	1
2	100	2
3	100	3
4	150	1
5	150	2
6	150	3
7	200	1
8	200	2
9	200	3
10	250	1
11	250	2
12	250	3
13	300	1
14	300	2
15	300	3

W trzecim zbiorze umieszczono fragmenty rzeczywistych sekwencji pobranych z GenBanku. Listę peptydów, które posłużyły do przygotowania danych, wraz z ich numerem w bazie GenBank, można znaleźć w załączniku A. W tym wypadku wykorzystano dwie rzeczywiste endopeptydazy: endoproteinase Lys-C, która rozpoznaje lizynę oraz rozcieńczony kwas, który przeprowadza cięcie po wystąpieniu asparaginy. W oparciu o rzeczywiste dane utworzono 15 podzbiorów sekwencji, z których każdy zawiera 10 z nich. Sekwencje zostały podzielone zgodnie z tabelą 7.3.

Tabela 7.3: Podział trzeciego zbioru peptydów na podzbiory.

Lp.	Długość sekwencji	Liczba błędów
1	100	1
2	100	2
3	100	3
4	150	1
5	150	2
6	150	3
7	200	1
8	200	2
9	200	3
10	250	1
11	250	2
12	250	3
13	300	1
14	300	2
15	300	3

Dla wszystkich sekwencji przeprowadzono **symulację eksperymentu chemicznego**. Zostało powiedziane, że brak cięcia nie jest źródłem typowego błędu pozytywnego lub negatywnego. Trawienie jest przeprowadzane na wielu cząsteczkach w roztworze równolegle. Brak cięcia w jednej takiej cząsteczce generuje dodatkową sekwencję, która pokrywa pewien fragment łańcucha. Brak tego samego cięcia we wszystkich cząsteczkach w roztworze generuje brak pewnej krótkiej sekwencji w spektrum.

Pierwsza błędna sytuacja, czyli powstanie dodatkowych łańcuchów, jest bardziej prawdopodobna, gdyż do powstania takiego łańcucha wystarczy jedynie, aby nie zaszło cięcie w dowolnej cząsteczce w roztworze.

Druga błędna sytuacja, czyli brak krótkich fragmentów jest bardzo mało prawdopodobna. Wymaga, aby to samo cięcie nie zaszło we wszystkich łańcuchach w roztworze. Na podstawie powyższych przemyśleń ustalono, że brak cięcia będzie się objawiał w symulacji powstaniem dodatkowych sekwencji, bardzo rzadko powodując utratę sekwencji, które powstają w przypadku idealnym (1% sytuacji). W pierwszym zbiorze instancji określa się liczbę **podstawień** w sekwencji sztucznych aminokwasów, które są rozpoznawalne przez endopeptydazę.

W drugim i trzecim zbiorze instancji podaje się procent błędów. Parametr ten określa ile procent dodatkowych sekwencji powstaje w eksperymencie w stosunku do przypadku idealnego. Należy podkreślić, że nawet w przypadku, gdy błędów jest 10% (czyli pojawia się 10% dodatkowych wierzchołków w stosunku do przypadku idealnego), rozważane instancje są bardzo trudne, ponieważ poza dodatkowymi wierzchołkami trzeba rozważyć wiele różnych nałożeń pomiędzy każdą parą sekwencji.

Komentarza może wymagać również wprowadzenie 1% sytuacji, w których nie pojawia się pewna krótka sekwencja w spektrum. Trzeba zauważyć, że wprowadzenie możliwości zagubienia krótkich sekwencji w spektrum było niezbędne, aby instancje nie odpowiadały problemowi „1.5", który jest łatwy obliczeniowo [27].

Wyniki symulacji eksperymentu chemicznego wraz z rozkładem aminokwasów w szukanej cząsteczce posłużyły jako dane wejściowe dla testowanych algorytmów. Zaproponowane algorytmy są metodami probabilistycznymi. Algorytm ewolucyjny zawiera probabilistyczny operator krzyżowania oraz w losowy sposób dobierane są osobniki w

procesie selekcji. W metodzie GRASP w losowy sposób dobierany jest wierzchołek z listy RLC oraz w losowy sposób przeszukiwane jest sąsiedztwo w procesie optymalizacji znalezionego rozwiązania początkowego. Algorytm Tabu również posiada elementy probabilistyczne. Z tego powodu każdy test został wykonany 10-krotnie a wyniki zostały uśrednione.

Przedstawione algorytmy bazują na tej samej funkcji oceny heurystycznej, minimalizującej różnicę pomiędzy szukanym rozkładem aminokwasów a rozkładem uzyskanym. Ocena jakości rozwiązania nie została jednak przeprowadzona w oparciu o uzyskane wartości funkcji oceny heurystycznej, tylko w oparciu o podobieństwo sekwencji uzyskanej do oryginalnej cząsteczki. Uzyskana wartość funkcji celu nie jest wystarczającą informacją na temat rzeczywistego podobieństwa do sekwencji oryginalnego łańcucha, gdyż dwa rozwiązania o tej samej wartości funkcji mogą być reprezentowane przez różne sekwencje.

Do porównania dwóch sekwencji wykorzystano algorytm Needlemana-Wunscha [43]. Algorytm oblicza podobieństwo sekwencji zgodnie z poniższym wzorem:

$$\sigma = 100\frac{\delta - \psi}{\chi - \psi}$$

gdzie δ to ocena dla dwóch sekwencji, która jest sumą punktów we wszystkich kolumnach dla optymalnego ustawienia sekwencji względem siebie (1 punkt jest przyznawany, gdy litery w kolumnie są takie same, -1 gdy litery się różnią lub jest luka). Poniżej podany jest wzór na ψ oraz χ:

$$\psi = \begin{cases} l(s_o)d + (l(s_t) - l(s_o))g & \text{gdy } l(s_t) > l(s_o) \\ l(s_t)d + (l(s_o) - l(s_t))g & \text{w przeciwnym wypadku} \end{cases}$$

$$\chi = \begin{cases} l(s_t)m & \text{gdy } l(s_t) > l(s_o) \\ l(s_o)m & \text{w przeciwnym wypadku} \end{cases}$$

gdzie $l(s_t)$ oraz $l(s_o)$ to długości sekwencji s_t and s_o, $m = 1$, $d = -1$, $g = -1$.

W tabelach 7.4 – 7.6 przedstawiono wyniki eksperymentu obliczeniowego dla metody GRASP [46] oraz algorytmu ewolucyjnego [7]. Dla każdego przypadku podano uśredniony wynik. Dopasowanie jest procentowym podobieństwem otrzymania sekwencji do sekwencji rzeczywistej, natomiast czas wykonywania to uśredniony czas zwrócenia przez algorytm wyniku dla danego podzbioru instancji. Wyniki skuteczniejszego algorytmu (ewolucyjnego) zestawiono z wynikami metody Tabu w tabelach 7.7 – 7.9.

Tabela 7.4: Wyniki algorytmu GRASP oraz algorytmu ewolucyjnego dla pierwszego zbioru instancji.

Długość sekwencji	Miejsca cięć	Błędy	GRASP		Algorytm Ewolucyjny	
			Podobieństwo [%]	Czas [s]	Podobieństwo [%]	Czas [s]
100	10	10%	72.50	1.17	98.15	0.98
100	10	20%	75.82	1.09	97.39	0.94
100	10	30%	81.83	1.09	96.86	1.08
100	15	10%	70.83	1.23	90.00	1.08
100	15	20%	70.15	1.09	90.46	1.10
100	15	30%	82.89	1.10	86.75	0.98
100	20	10%	90.16	1.04	78.33	0.94
100	20	20%	71.09	1.02	74.60	1.08

Długość sekwencji	Miejsca cięć	Błędy	GRASP		Algorytm Ewolucyjny	
			Podobieństwo [%]	Czas [s]	Podobieństwo [%]	Czas [s]
100	20	30%	78.13	1.09	78.26	1.09
150	10	10%	90.09	1.10	93.45	0.96
150	10	20%	77.09	1.02	88.83	1.08
150	10	30%	71.15	1	98.86	1.10
150	15	10%	82.16	1.01	90.45	1.20
150	15	20%	73.76	1.05	92.79	1.02
150	15	30%	70.02	1.11	90.76	1.06
150	20	10%	70.01	1.10	89.05	1.14
150	20	20%	69.19	1.23	91.33	1.06
150	20	30%	65.59	0.97	91.67	1.05
200	10	10%	71.89	1.01	98.10	1.12
200	10	20%	84.89	1.02	98.47	1.09
200	10	30%	94.13	1.12	100.00	1.09
200	15	10%	72.08	1.21	88.52	1.05
200	15	20%	61.13	1.20	91.68	1.09
200	15	30%	73.45	1.20	92.66	1.06
200	20	10%	74.87	1.19	91.01	1.27
200	20	20%	66.72	1.32	91.38	1.15
200	20	30%	81.13	1.08	93.96	1.00
250	10	10%	78.15	1.22	100.00	1.33
250	10	20%	75.6	1.1	93.40	1.05
250	10	30%	87.91	1.09	97.50	1.01
250	15	10%	72.48	1.10	92.09	1.18
250	15	20%	72.74	1.10	89.58	1.11
250	15	30%	80.01	1.26	99.47	1.03
250	20	10%	75.28	1.14	82.35	1.09
250	20	20%	72.16	1.09	90.73	1.16
250	20	30%	69.08	1.21	98.00	1.16
300	10	10%	81.03	1.20	92.77	1.24
300	10	20%	80.98	1.17	100.00	1.08
300	10	30%	83.34	1.45	98.98	1.25
300	15	10%	67.70	1.18	96.59	1.19
300	15	20%	64.18	1.41	95.30	1.01
300	15	30%	73.17	1.66	96.37	1.10
300	20	10%	78.24	1.70	90.52	1.28
300	20	20%	76.10	1.84	95.32	1.06
300	20	30%	73.22	1.77	95.26	1.09

Tabela 7.5: Wyniki algorytmu GRASP oraz algorytmu ewolucyjnego dla drugiego zbioru instancji.

Długość sekwencji	Liczba błędów	GRASP		Algorytm Ewolucyjny	
		Podobieństwo [%]	Czas [s]	Podobieństwo [%]	czas [s]
100	1	70.13	0.99	86.46	1.92
100	2	64.48	1.02	88.18	2.29
100	3	63.16	1.05	92.36	2.68

Długość sekwencji	Liczba błędów	GRASP Podobieństwo [%]	GRASP Czas [s]	Algorytm Ewolucyjny Podobieństwo [%]	Algorytm Ewolucyjny czas [s]
150	1	66.40	1.20	80.58	2.58
150	2	69.11	0.98	84.70	2.44
150	3	64.24	1.01	87.62	2.12
200	1	62.27	1.13	80.16	2.93
200	2	68.11	1.08	79.78	1.76
200	3	61.02	1.08	81.53	2.97
250	1	63.1	1.093	79.13	2.07
250	2	62.75	0.92	80.24	1.76
250	3	59.99	1.14	81.60	2.18
300	1	59.44	1.18	81.05	3.00
300	2	63.19	1.7	78.86	2.93
300	3	58.15	1.2	80.46	2.69

Tabela 7.6: Wyniki algorytmu GRASP oraz algorytmu ewolucyjnego dla trzeciego zbioru instancji.

Długość sekwencji	Liczba błędów	GRASP Podobieństwo [%]	GRASP Czas [s]	Algorytm Ewolucyjny Podobieństwo [%]	Algorytm Ewolucyjny Czas [s]
100	1	82.17	0.87	82.58	1.87
100	2	87.56	0.92	87.19	1.93
100	3	89.17	0.92	88.92	1.89
150	1	85.94	1.08	83.54	1.73
150	2	90.17	1.00	81.17	2.02
150	3	75.38	1.14	85.67	1.98
200	1	74.95	1.13	80.19	1.86
200	2	81.02	1.12	81.12	1.92
200	3	72.17	1.10	84.56	2.01
250	1	63.51	1.44	81.86	1.89
250	2	63.14	1.49	79.34	1.99
250	3	63.95	1.40	83.27	1.94
300	1	65.18	1.64	83.40	1.96
300	2	69.29	1.60	80.96	2.02
300	3	60.93	1.57	81.02	2.11

Tabela 7.7: Wyniki metody Tabu oraz algorytmu ewolucyjnego dla pierwszego zbioru instancji.

Długość sekwencji	Miejsca cięć	Błędy	Metoda Tabu Podobieństwo [%]	Metoda Tabu Czas [s]	Algorytm Ewolucyjny Podobieństwo [%]	Algorytm Ewolucyjny Czas [s]
100	10	10%	77.20	2.39	98.15	0.98
100	10	20%	80.80	1.05	97.39	0.94
100	10	30%	78.70	1.29	96.86	1.08
100	15	10%	55.83	8.82	90.00	1.08
100	15	20%	56.41	5.93	90.46	1.10
100	15	30%	60.42	5.39	86.75	0.98
100	20	10%	40.41	38.01	78.33	0.94
100	20	20%	41.77	25.41	74.60	1.08

Długość sekwencji	Miejsca cięć	Błędy	Metoda Tabu		Algorytm Ewolucyjny	
			Podobieństwo [%]	Czas [s]	Podobieństwo [%]	Czas [s]
100	20	30%	49.78	14.91	78.26	1.09
150	10	10%	65.39	6.85	93.45	0.96
150	10	20%	73.85	4.30	88.83	1.08
150	10	30%	85.90	1.85	98.86	1.10
150	15	10%	48.11	19.11	90.45	1.20
150	15	20%	55.64	14.47	92.79	1.02
150	15	30%	53.80	10.79	90.76	1.06
150	20	10%	46.33	41.71	89.05	1.14
150	20	20%	55.00	23.99	91.33	1.06
150	20	30%	50.08	22.41	91.67	1.05
200	10	10%	66.62	4.73	98.10	1.12
200	10	20%	84.89	4.44	98.47	1.09
200	10	30%	95.44	1.40	100.00	1.09
200	15	10%	50.76	22.16	88.52	1.05
200	15	20%	57.38	10.26	91.68	1.09
200	15	30%	60.81	12.78	92.66	1.06
200	20	10%	42.90	68.99	91.01	1.27
200	20	20%	49.50	40.03	91.38	1.15
200	20	30%	52.79	22.31	93.96	1.00
250	10	10%	85.52	3.37	100.00	1.33
250	10	20%	74.30	5.62	93.40	1.05
250	10	30%	83.53	3.28	97.50	1.01
250	15	10%	59.89	14.50	92.09	1.18
250	15	20%	55.97	15.61	89.58	1.11
250	15	30%	72.20	8.63	99.47	1.03
250	20	10%	45.25	66.94	82.35	1.09
250	20	20%	43.51	62.48	90.73	1.16
250	20	30%	60.14	17.48	98.00	1.16
300	10	10%	76.67	11.69	92.77	1.24
300	10	20%	76.23	4.36	100.00	1.08
300	10	30%	84.23	3.44	98.98	1.25
300	15	10%	59.16	15.29	96.59	1.19
300	15	20%	63.19	12.88	95.30	1.01
300	15	30%	69.71	10.13	96.37	1.10
300	20	10%	48.12	56.94	90.52	1.28
300	20	20%	59.25	23.33	95.32	1.06
300	20	30%	50.93	28.62	95.26	1.09

Tabela 7.8: Wyniki metody Tabu oraz algorytmu ewolucyjnego dla drugiego zbioru instancji.

Długość sekwencji	Liczba błędów	Metoda Tabu		Algorytm Ewolucyjny	
		Podobieństwo [%]	Czas [s]	Podobieństwo [%]	czas [s]
100	1	62,55	6.35	86.46	1.92
100	2	70.86	5.76	88.18	2.29
100	3	73.13	3.80	92.36	2.68

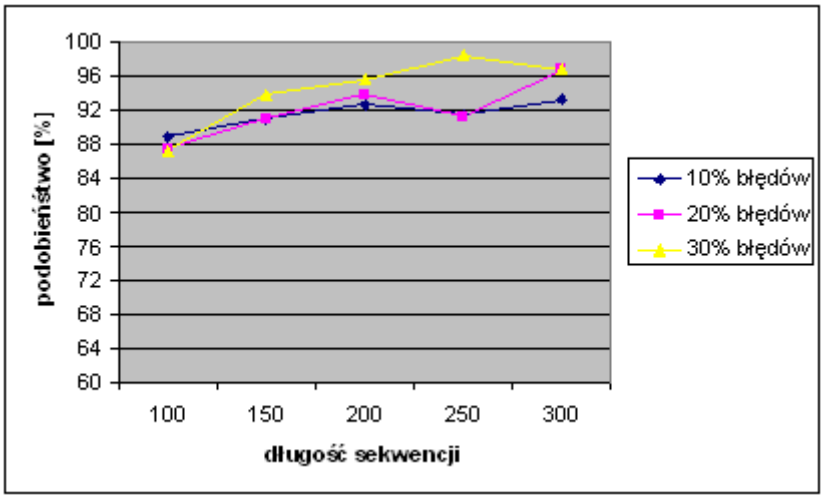
Długość sekwencji	Liczba błędów	Metoda Tabu		Algorytm Ewolucyjny	
		Podobieństwo [%]	Czas [s]	Podobieństwo [%]	czas [s]
150	1	46.72	21.44	80.58	2.58
150	2	53.52	16.23	84.70	2.44
150	3	56.77	9.05	87.62	2.12
200	1	46.36	84.31	80.16	2.93
200	2	43.22	77.04	79.78	1.76
200	3	48.22	64.15	81.53	2.97
250	1	41.36	200.60	79.13	2.07
250	2	43.42	167.90	80.24	1.76
250	3	44.10	147.35	81.60	2.18
300	1	40.53	384.89	81.05	3.00
300	2	40.64	366.26	78.86	2.93
300	3	41.12	197.10	80.46	2.69

Tabela 7.9: Wyniki metody Tabu oraz algorytmu ewolucyjnego dla trzeciego zbioru instancji.

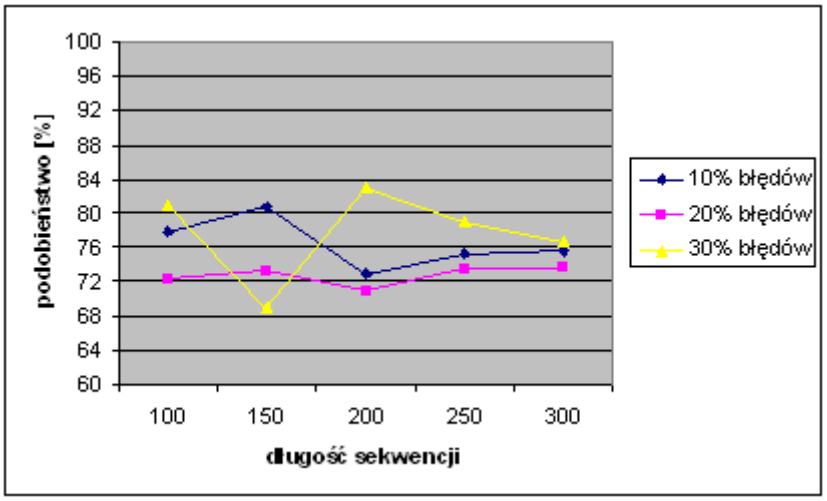
Długość sekwencji	Liczba błędów	Metoda Tabu		Algorytm Ewolucyjny	
		Podobieństwo [%]	Czas [s]	Podobieństwo [%]	Czas [s]
100	1	81.33	6.07	82.58	1.87
100	2	85.73	1.76	87.19	1.93
100	3	65.66	2.11	88.92	1.89
150	1	70.22	15.39	83.54	1.73
150	2	73.89	14.73	81.17	2.02
150	3	74.19	12.62	85.67	1.98
200	1	56.67	86.49	80.19	1.86
200	2	58.91	58.58	81.12	1.92
200	3	62.41	51.43	84.56	2.01
250	1	49.22	150.46	81.86	1.89
250	2	49.54	154.40	79.34	1.99
250	3	54.23	93.04	83.27	1.94
300	1	45.88	289.12	83.40	1.96
300	2	47.98	289.48	80.96	2.02
300	3	44.58	250.75	81.02	2.11

Eksperyment obliczeniowy pokazuje, że zarówno metoda GRASP jak i algorytm zachłanny dają lepsze wyniki niż metoda Tabu. Uśrednione wartości wynoszą 59,7% dla metody Tabu, 73,32% dla metody GRASP oraz 88,74% dla algorytmu ewolucyjnego. Wyniki przeprowadzonych eksperymentów obliczeniowych zostały zobrazowane w postaci wykresów przedstawionych na rysunkach 7.1 - 7.12. Eksperyment obliczeniowy rozpoczęto od zaimplementowania i przetestowania metody Tabu oraz algorytmu Ewolucyjnego [7]. Algorytm ewolucyjny uzyskał lepsze wyniki dla prawie wszystkich testów niż metoda Tabu. Uzyskane przez ten algorytm sekwencje są jest średnio o 29.04% lepiej dopasowane do szukanej sekwencji niż te uzyskane za pomocy metody Tabu. Powodem tego może być fakt, że wykorzystany przez ten algorytm ewolucyjny operator krzyżowania (Algorithm 3) znacznie preferuje te rozwiązania, które odpowiadają sytuacji, kiedy sąsiadujące ze sobą krótkie peptydy nakładają się. Odpowiada to sytuacji rzeczywistej,

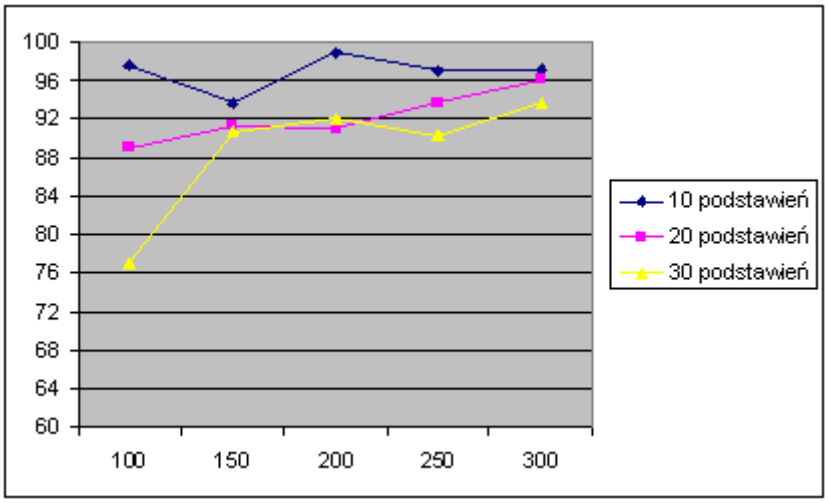
gdzie w znalezionej permutacja krótkich sekwencji każde dwa sąsiadujące sekwencje powinny się na siebie częściowo nakładać. Podczas gdy algorytm ewolucyjny, poprzez zastosowanie operatora krzyżowania, uwzględnia tą właściwość podczas budowania rozwiązań, metoda Tabu nie korzysta z żadnych dodatkowych własności problemu. Kolejną zaimplementowaną i przetestowaną metodą była metaheurystyka GRASP, której uśrednione wyniki są średnio o 13.62% lepsze niż Tabu. Podczas budowania rozwiązania początkowego, metoda ta preferuje takie elementy, których dodanie do częściowego rozwiązania odpowiada sytuacji ustawienia obok siebie w rozwiązaniu dwóch krótkich sekwencji, które częściowo się nakładają. Rozwiązanie początkowe w metodzie Tabu jest znajdowane losowo i nie uwzględnia spostrzeżeń wykorzystanych przy projektowaniu GRASP. Może to w pewien sposób tłumaczyć lepsze wyniki otrzymane przez tą drugą metodę.



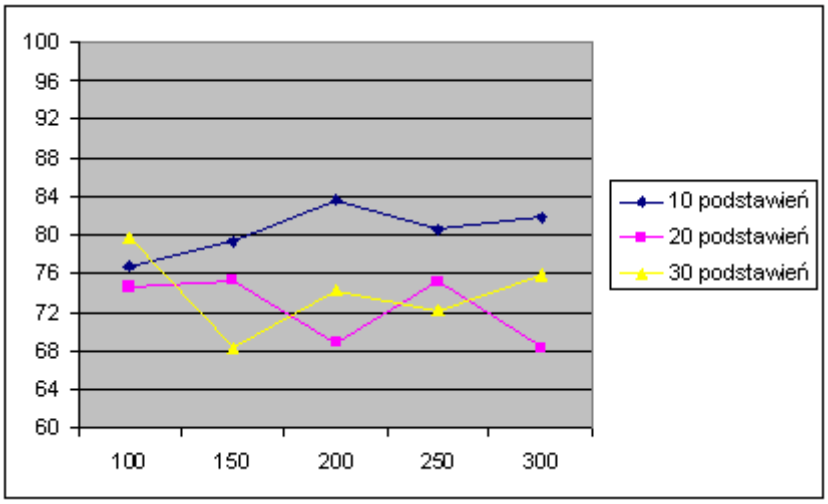
Rysunek 7.1: Uśrednione wyniki algorytmu ewolucyjnego dla pierwszego zbioru instancji z podziałem na liczbę błędów



Rysunek 7.2: Uśrednione wyniki metody GRASP dla pierwszego zbioru instancji z podziałem na liczbę błędów



Rysunek 7.3: Uśrednione wyniki algorytmu ewolucyjnego dla pierwszego zbioru instancji z podziałem na liczbę podstawień



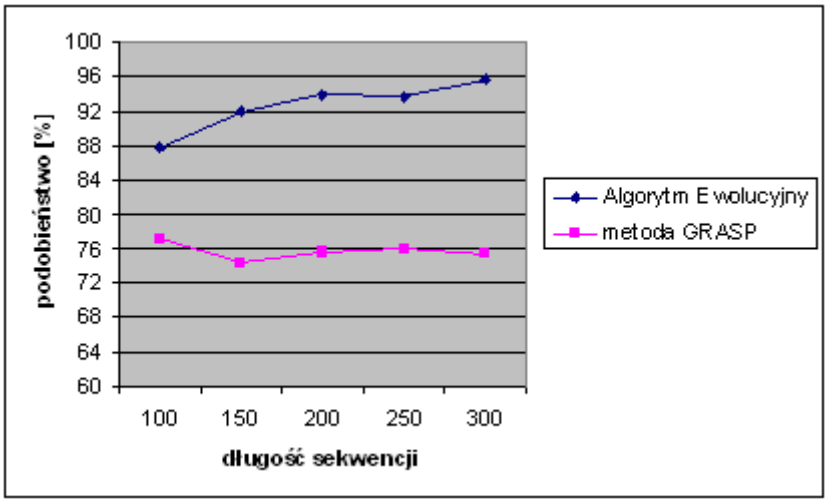
Rysunek 7.4: Uśrednione wyniki metody GRASP dla pierwszego zbioru instancji z podziałem na liczbę podstawień

7.3.1 Heurystyka dedykowana

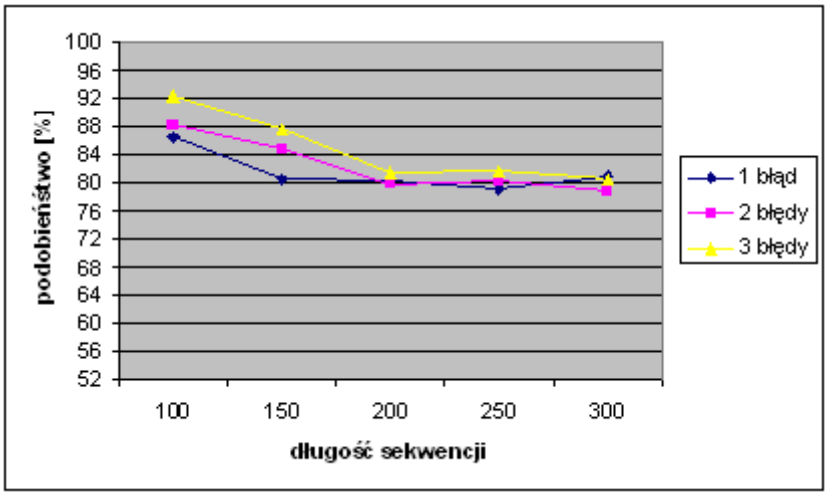
Eksperyment obliczeniowy dla heurystyki dedykowanej został przeprowadzony dla rzeczywistych sekwencji białkowych. Wykorzystano trzeci zbiór sekwencji z eksperymentu dla metaheurystyk.

W przypadku metaheurystyk liczba błędów w procesie eksperymentu chemicznego ma niewielki wpływ na rezultaty. Ważny jest fakt występowania błędów, gdyż powoduje, że nie można skorzystać z modelu grafów peptydowych - problem staje się trudny obliczeniowo.

W przypadku heurystyki dedykowanej, liczba błędów ma duże znaczenie. Dla instancji z 1, 2 albo 3 błędami przestrzeń rozwiązań dla heurystyki jest tak niewielka, że można znaleźć optymalne rozwiązanie dla tych instancji w czasie 1 sekundy.



Rysunek 7.5: Porównanie algorytmu ewolucyjnego i metody GRASP dla pierwszego zbioru instancji



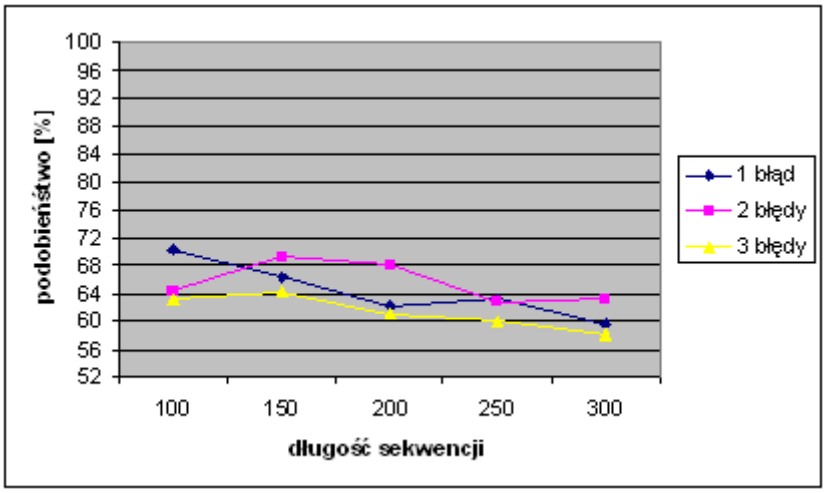
Rysunek 7.6: Uśrednione wyniki algorytmu ewolucyjnego dla drugiego zbioru instancji z podziałem na liczbę błędów

W celu przetestowania możliwości heurystyki dedykowanej w trakcie symulacji eksperymentu chemicznego wprowadzono kolejno 10%, 20% oraz 30% błędów, co indukuje dodatkowe 10 do 30 procent wierzchołków w grafie.

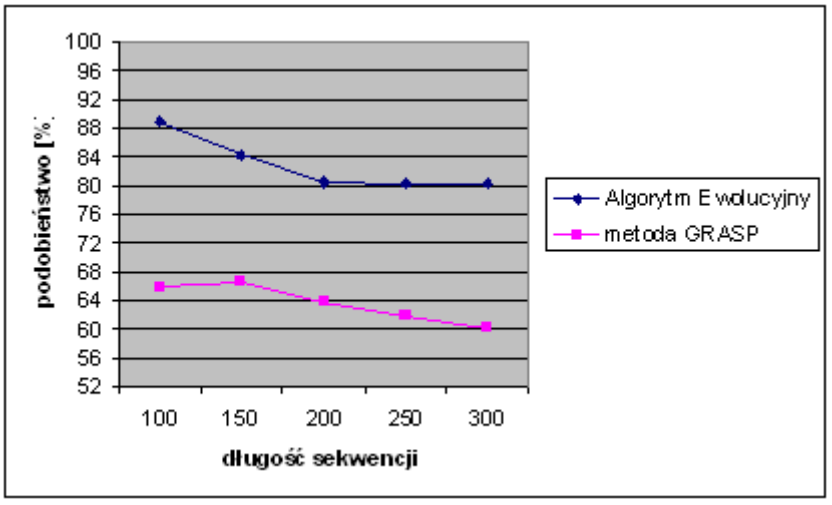
Algorytm został wykonany 10-krotnie dla każdej sekwencji, a uzyskane wyniki uśredniono. Czas działania metody został ograniczony do 1 sekundy. W tabeli 7.10 oraz na rysunku 7.13 przedstawiono wyniki eksperymentu. Uśrednione wyniki wynoszą $96,07\%$ i znacznie przewyższają rezultaty jakiegokolwiek przypadku testowego dla metaheurystyk. Trzeba jednak podkreślić, że analiza metody Tabu oraz badania nad algorytmem ewolucyjnym i metodą GRASP należy traktować jako kolejne kroki prowadzące do zaproponowania tak dobrej metody dedykowanej.

Tabela 7.10: Wyniki heurystyki dedykowanej.

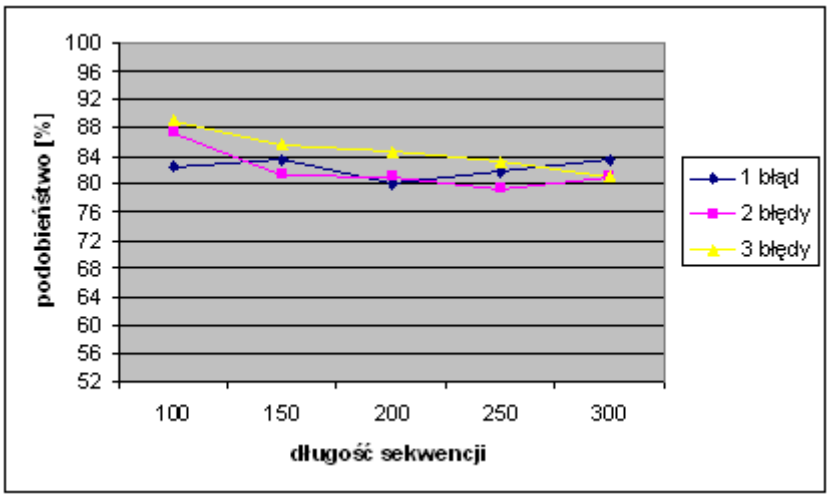
Długość sekwencji	Błędy	Podobieństwo [%]
Długość sekwencji	Błędy	Podobieństwo [%]
100	1	99.34
100	2	99.01
100	3	98.18
150	1	97.49
150	2	98.60
150	3	93.28
200	1	97.86
200	2	94.09
200	3	92.81
250	1	96.41
250	2	95.97
250	3	92.01
300	1	97.01
300	2	95.81
300	3	93.11



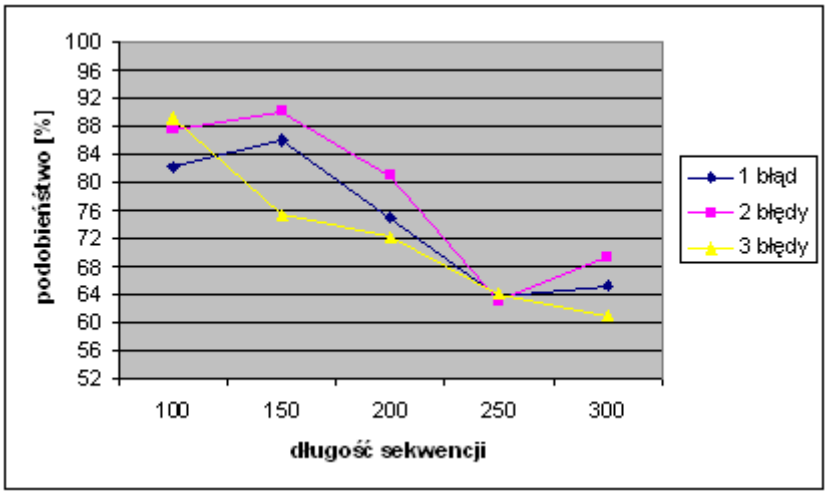
Rysunek 7.7: Uśrednione wyniki metody GRASP dla drugiego zbioru instancji z podziałem na liczbę błędów



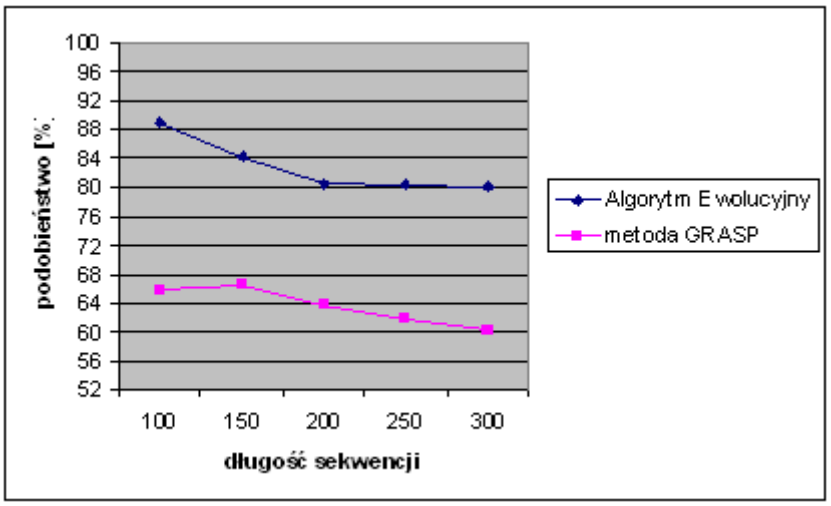
Rysunek 7.8: Porównanie algorytmu ewolucyjnego i metody GRASP dla drugiego zbioru instancji



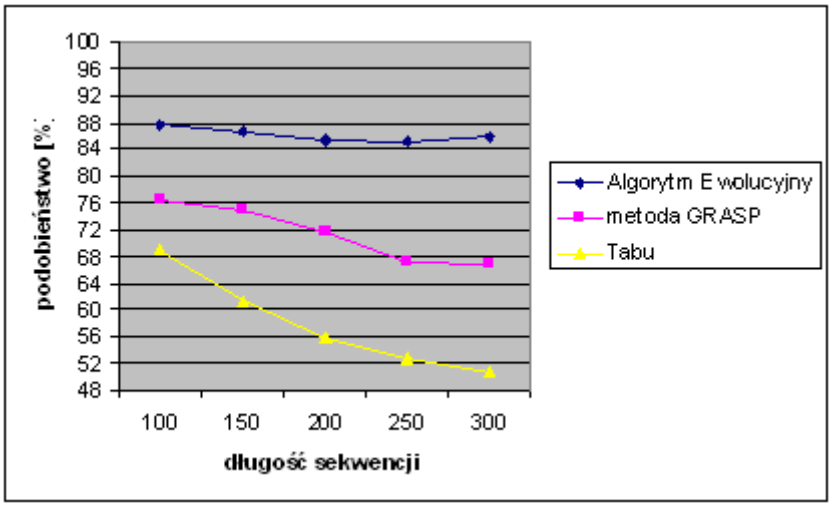
Rysunek 7.9: Uśrednione wyniki algorytmu ewolucyjnego dla trzeciego zbioru instancji z podziałem na liczbę błędów



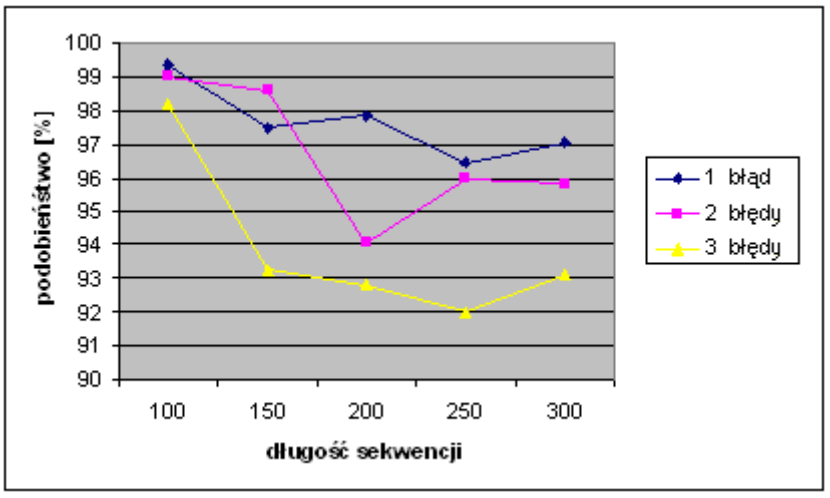
Rysunek 7.10: Uśrednione wyniki metody GRASP dla trzeciego zbioru instancji z podziałem na liczbę błędów



Rysunek 7.11: Porównanie algorytmu ewolucyjnego i metody GRASP dla trzeciego zbioru instancji



Rysunek 7.12: Uśrednione wyniki wszystkich metod



Rysunek 7.13: Wyniki dla heurystyki dedykowanej

Rozdział 8

Podsumowanie

Ostatnie dwie dekady to okres szybkiego rozwoju nauk z pogranicza biologii, matematyki i informatyki. Często rozwiązanie problemów biologicznych wymaga zaangażowania nowoczesnych maszyn obliczeniowych i zaprojektowania specjalistycznych narzędzi informatycznych. Szczególnie interesujące jest badanie sekwencji biologicznych: białek i DNA. Kwas deoksyrybonukleinowy jest nośnikiem informacji genetycznej a białko określa fenotyp organizmu. Największym osiągnięciem ostatnich lat w biologii sekwencji jest ustalenie sekwencji genomu ludzkiego. Nieustannie powstają nowe podejścia, metody i narzędzia wspomagające ustalanie budowy, funkcji i zastosowań cząsteczek biologicznych. Ta praca doktorska wpasowuje się w te trendy.

W pracy przedstawiono klasyfikację problemów sekwencjonowania oraz problemów asemblacji peptydów.

W pracy opisano klasę grafów peptydowych, które są reprezentacją problemu asemblacji peptydów bez błędów. Zaproponowano algorytm, który w wielomianowym czasie weryfikuje czy dany graf należy do tej klasy. Udowodniono, że problem asemblacji ze wszystkimi cięciami i danym rozkładem aminokwasów jest łatwy obliczeniowo. W ten sposób pokazano, że dodanie informacji o rozkładzie nie zmienia klasy złożoności problemu asemblacji. To ważny wniosek - wskazówka dla biochemików, że dostarczenie tej dodatkowej informacji na wejściu dla metody asemblacyjnej nie wpłynie na znaczne wydłużenie czasu działania programu, a może polepszyć jakość otrzymanych rozwiązań.

Zaproponowano szereg metod do kompleksowego określenia sekwencji białkowych. Przedstawiono metodę sekwencjonowania de novo. Dla problemu asemblacji trudnego obliczeniowo zaproponowano i zaimplementowano trzy metody przybliżone. Przeprowadzono eksperyment obliczeniowy w celu oceny tych metod. Na szczególną uwagę zasługuję heurystyka dedykowana, w wyniku działania której otrzymuje się sekwencje, których podobieństwo do oryginalnej sekwencji przekracza 96%.

W rozprawie zaproponowano algorytm budowania bibliotek oligonukleotydów, których wzajemna zdolność do wzajemnej hybrydyzacji jest minimalna. Biblioteki takie mogą zostać wykorzystane do optymalizacji sposobu kodowania instancji będących danymi wejściowymi dla algorytmów przeznaczonych do wykonania za pomocą komputerów DNA.

Analizy przeprowadzone na potrzeby tej rozprawy wskazały nowe, przyszłe potencjalne obszary badań:

- przeprowadzenie eksperymentu obliczeniowego w celu oceny wpływu zaproponowanej metody budowy bibliotek na wydajność algorytmów przeznaczonych do wykonania za pomocą komputerów DNA;

- zaproponowanie metody dokładnej dla problemu asemblacji z błędami i znanym rozkładem,
- przeprowadzenie eksperymentu obliczeniowego dla zaproponowanej heurystyki dedykowanej dla dłuższych sekwencji; dalsza optymalizacja metody,
- zaimplementowanie i przetestowanie opisanej metody sekwencjonowania de novo w celu jej porównania z innymi rozwiązaniami dostępnymi w literaturze,
- określenie złożoności obliczeniowej dla problemów asemblacji, w przypadku gdy nie ma informacji o powtórzeniach.

Dodatek A

Zasoby

A.1 Lista sekwencji peptydowych wykorzystanych w eksperymencie obliczeniowym wraz z ich kodem dostępowym w GenBank

AAI22861 SARM1 protein [Homo sapiens]

AAI22867 PTPRR protein [Homo sapiens]

AAI25270 Chromosome 16 open reading frame 84 [Homo sapiens]

AAI25271 FSHR protein [Homo sapiens]

AAI25273 gi|115940697|gb|AAI25273.1|

AAI25274 ZNF497 protein [Homo sapiens]

ABI97387 ADAM metallopeptidase domain 33 [Homo sapiens]

ABI97388 ATP-binding cassette, sub-family G (WHITE), member 2 [Homo sapiens]

ABI98401 lung specific F-box and DH domain containing protein [Homo sapiens]

ABJ09587 sodium-driven chloride bicarbonate exchanger [Homo sapiens]

A.2 Modyfikacje potranslacyjne sekwencji aminokwasowych

Tabela A.1: Zestawienie znanych modyfikacji potranslacyjnych sekwencji aminokwasowych

Modyfikacja	Zmiana masy cząsteczki [Da]
5' dephospho	-79
Desmosine (from Lysine)	-58
decomposed carboxymethylated Methionine	-48
decarboxylation of gamma carboxy Glutamate	-44
gamma-glutamyl semialdehyde (from arginine)	-43
Ornithine (from Arginine)	-42
Lysinoalanine (from Cysteine)	-34
Lanthionine (from Cysteine)	-34
Dehydroalanine (from Cysteine)	-34

Modyfikacja	Zmiana masy cząsteczki [Da]
Homoserine formed from Met by CNBr treatment	-30
Oxidation of arginine (to glutamic acid)	-27
Formylglycine (from cysteine)	-18
Pyroglutamic Acid formed from Glutamic Acid	-18
Dehydration (-H2O)	-18
S-gamma-Glutamyl (crosslinked to Cysteine)	-18
O-gamma-Glutamyl- (Crosslink to Serine)	-18
Serine to Dehydroalanine	-18
Alaninohistidine (Serine crosslinked to theta or pi carbon of Histidine)	-18
Misincorporation of Norleucine for Methionine	-18
Succinimide formation from aspartic acid	-18
Pyroglutamic Acid formed from Gln	-17
N-pyrrolidone carboxyl (N terminus)	-17
N alpha -(gamma-Glutamyl)-lysine	-17
N-(beta-Aspartyl)-Lysine (Crosslink)	-17
Succinimide formation from asparagine	-17
S-carbamoylmethylcysteine cyclization (N-terminus)	-17
Pyruvoyl- (Serine)	-16
Crosslink between Arg and His sidechains	-5
3,3',5,5'-TerTyr (Crosslink)	-4
Formylglycine (from serine)	-2
Disulphide bond formation (Cystine)	-2
S-(2-Histidyl)- (Crosslinked to Cysteine)	-2
S-(3-Tyr) (Crosslinked to Cysteine)	-2
3,3'-BiTyr (Crosslink)	-2
IsodiTyr (Crosslink)	-2
Allysine (from Lysine)	-1
Amide formation (C terminus)	-1
Oxidation of lysine (to aminoadipic semialdehyde)	-1
Deamidation of Asparagine and Glutamine to Aspartate and Glutamate	1
Citruline (from Arginine)	1
Cysteine x2, reduction of Cystine (Cys-Cys)	2
Reduction of indole double bond of Trp	2
Oxidation of Trp to kynurenine	4
Lysine epsilon amino to imine	12
Cysteine (N-term) formaldehyde adduct (Cys to Thioproline conversion)	12
Formaldehyde adduct of Trp	12
Syndesine (from Lysine)	13
CM-Cys vs PAM-Cys	13
Methylation (N terminus, N epsilon of Lysine, O of Serine, Threonine or C terminus, N of Asparagine)	14
CAM-Cys vs PAM-Cys	14
delta-Hydroxy-allysine (from Lysine)	15
Oxidation of lysine (to aminoadipic acid)	15

Modyfikacja	Zmiana masy cząsteczki [Da]
Hydroxylation (of delta C of Lysine, beta C of Tryptophan, C3 or C4 of Proline, beta C of Aspartate)	16
Oxidation of Methionine (to Sulphoxide)	16
3,4-Dihydroxy-Phenylalanine (from Tyrosine) (DOPA)	16
Oxohistidine (from histidine)	16
Sulfenic Acid (from Cysteine)	16
Oxidation of proline (to gamma-glutamyl semialdehyde)	16
Sodium	22
Ethyl	28
N,N dimethylation (of Arginine or Lysine)	28
2,4-BisTrp-6,7-dione (from Tryptophan)	28
Formylation (CHO)	28
6,7 Dione (from Tryptophan)	30
3,4,6-Trihydroxy-Phenylalanine (from Tyrosine) (TOPA)	32
3,4-Dihydroxylation (of Proline)	32
Oxidation of Methionine (to Sulphone)	32
Oxidation of proline (to glutamic acid)	32
Double oxidation of Trp	32
3-Chlorination (of Tyrosine with 35Cl)	34
3-Chlorination (of Tyrosine with 37Cl)	36
Potassium	38
Acetylation (N terminus, N epsilon of Lysine, O of Serine) (Ac)	42
N-Trimethylation (of Lysine)	42
Carbamylation	43
disodium	44
Nitro (NO2)	45
beta-Methylthio-aspartic acid	46
Cysteic acid, oxidation of cysteine	48
Piperidine adduct to C-terminal Cys	51
t-butyl ester(OtBu) and t-butyl (tBu)	56
Glycyl (-G-,-Gly-)	57
Carboxamidomethyl (on Cysteine)	57
Carboxymethyl (on Cysteine)	58
sodium + potassium	60
Selenocysteine (from Serine)	64
Asp transamidation with piperidine	67
3,5-Dichlorination (of Tyrosine with 35Cl)	68
Dehydroalanine (Dha)	69
3,5-Dichlorination (of Tyrosine with mixture of 35Cl and 37Cl))	70
Pyruvate	70
Sarcosyl	71
Alanyl (-A-, -Ala-)	71
Acetamidomethyl (Acm)	71
Propionamide or Acrylamide adduct	71
3,5-Dichlorination (of Tyrosine with 37Cl)	72
S-(sn-1-Glyceryl) (on Cysteine)	74
Glycerol Ester (on Glutamic acid side chain)	74
Glycine (G, Gly)	75

Modyfikacja	Zmiana masy cząsteczki [Da]
Phenyl ester (OPh) (on acidic)	76
Beta mercaptoethanol adduct	76
3-Bromination (of Tyrosine with 79Br)	78
L-o-bromination of Phe with 79Br	78
L-o-bromination of Phe with 81Br	80
Sulphonation (SO3H) (of PMC group)	80
Sulphation (of O of Tyrosine)	80
Phosphorylation (O of Serine, Threonine, Tyrosine and Aspartate, N epsilon of Lysine)	80
3-Bromination (of Tyrosine with 81Br)	80
Cyclohexyl ester (OcHex)	82
Dehydroamino butyric acid (Dhb)	83
Homoseryl lactone	83
2-Aminobutyric acid (Abu)	85
2-Aminoisobutyric acid (Aib)	85
Gamma Aminobutyryl	85
t-butyloxymethyl (Bum)	86
Diaminopropionyl	86
N-(4-NH2-2-OH-butyl)- (of Lysine) (Hypusine)	87
Seryl (-S-, -Ser-)	87
t-butylsulfenyl (StBu)	88
Alanine (A, Ala)	89
Sarcosine (Sar)	89
Anisyl	90
Benzyl (Bzl) and benzly ester (OBzl)	90
1,2-ethanedithiol (EDT)	93
Dehydroprolyl	95
Trifluoroacetyl (TFA)	96
N-hydroxysuccinimide (ONSu, OSu)	97
Prolyl (-P-, -Pro-)	97
Cysteic acid x2, oxidation of cystine	98
Tetramethylguanidinium termination by-product on amine	98
Phosphate/sulphate adduct of proteins	98
Valyl (-V-, -Val-)	99
Isovalyl (-I-,-Iva-)	99
t-Butyloxycarbonyl (tBoc)	100
Threoyl (-T-, -Thr-)	101
Homoseryl (-Hse-)	101
Cystyl (-C-, -Cys-)	103
4-Methylbenzyl (Meb)	104
Benzoyl (Bz)	104
Serine (S, Ser)	105
Pyridylethylation of cysteine	105
HMP (hydroxymethylphenyl) linker	106
Thioanisyl	106
Thiocresyl	106
2-Piperidinecarboxylic acid (Pip)	111
Pyroglutamyl	111

Modyfikacja	Zmiana masy cząsteczki [Da]
Hydroxyprolyl (-Hyp-)	113
Isoleucyl (-I-, -Ile-)	113
Leucyl (-L-, -Leu-)	113
Norleucyl (-Nle-)	113
Asparagyl (-N-, -Asn-)	114
t-amyloxycarbonyl (Aoc)	114
Ornithyl (-Orn-)	114
Proline (P, Pro)	115
Aspartyl (-D-, -Asp-)	115
Succinyl	117
Valine (V, Val)	117
Hydroxybenzotriazole ester (HOBt)	117
Dimethylbenzyl (diMeBzl)	118
Threonine (T, Thr)	119
Cysteinylation	119
Benzyloxymethyl (Bom)	120
p-methoxybenzyl (Mob, Mbzl)	120
4-Nitrophenyl, p-Nitrophenyl (ONp)	121
Cysteine (C, Cys)	121
Chlorobenzyl (ClBzl)	125
Iodination (of Histidine[C4] or Tyrosine[C3])	126
octanoylation	126
Glutamyl (-Q-, -Gln-)	128
Lysyl (-K-, -Lys-)	128
Glutamyl (-E-, -Glu-)	129
O-Methyl Aspartamyl	129
N alpha -(gamma-Glutamyl)-Glu	130
Norleucine (Nle)	131
Hydroxyproline (Hyp)	131
Isoleucine (I, Ile)	131
Leucine (L, Leu)	131
Methionyl (-M-, -Met-)	131
Hydroxy Aspartamyl	131
bb-dimethyl Cystenyl	131
Asparagine (N, Asn)	132
Pentoses (Ara, Rib, Xyl)	132
Aspartic Acid (D, Asp)	133
Benzyloxycarbonyl (Z)	134
Adamantyl (Ada)	134
p-Nitrobenzyl ester (ONb)	135
Histidyl (-H-, -His-)	137
N-methyl Glutamyl	142
N-methyl Lysyl	142
O-methyl Glutamyl	143
Diphthamide (from Histidine)	143
Hydroxy Lysyl (-Hyl-)	144
Methyl Methionyl	145
Glutamine (Q, Gln)	146

Modyfikacja	Zmiana masy cząsteczki [Da]
Deoxyhexoses (Fuc, Rha)	146
Lysine (K, Lys)	146
Pentosyl	146
Aminoethyl Cysteinyl (AECys)	146
4-Glycosyloxy- (pentosyl,C5) (of Proline)	147
Glutamic Acid (E, Glu)	147
Phenylalanyl- (-F-, -Phe-)	147
Methionyl Sulfoxide	147
Pyridyl Alanyl	148
2-Nitrobenzoyl (NBz)	149
Methionine (M, Met)	149
Fluorophenylalanyl	149
Dimethoxybenzyl Trp	150
2-Nitrophenylsulphenyl (Nps)	153
4-Toluenesulphonyl (Tosyl, Tos)	154
3-nitro-2-pyridinesulfenyl (Npys)	154
Histidine (H, His)	155
3,5-Dibromination (of Tyrosine with 79Br)	156
Arginyl (-R-, -Arg-)	156
Citrulline	157
3,5-Dibromination (of Tyrosine with mixture of 79Br and 81Br)	158
Dichlorobenzyl (Dcb)	159
3,5-Dibromination (of Tyrosine with 81Br)	160
Carboxyamidomethyl Cystenyl	160
Hexosamines (GalN, GlcN)	161
Carboxymethyl cysteine (Cmc)	161
Carboxymethyl Cystenyl	161
Methylphenylalanyl	161
Inositol	162
N-Glucosyl (N terminus or N epsilon of Lysine) (Aminoketose)	162
O-Glycosyl- (to Serine or Threonine)	162
Linker attached to peptide in Fmoc peptide synthesis	162
Hexoses (Fru, Gal, Glc, Man)	162
Tyrosinyl (-Y-, -Tyr-)	163
MethionylSulphone	163
Phenylalanine (F, Phe)	165
2,4 -dinitrophenyl (Dnp)	166
Pentaflourophenyl (Pfp)	166
Diphenylmethyl (Dpm)	166
Phospho Seryl	167
2-Chlorobenzyloxycarbonyl (ClZ)	169
Napthyl acetyl	169
N-acetyl Lysyl	170
N-methyl Arginyl	170
Ethanedithiol/TFA cyclic adduct	172
Carboxy Glutamyl (Gla)	173
Arginine (R, Arg)	174
Acetamidomethyl Cystenyl	174

Modyfikacja	Zmiana masy cząsteczki [Da]
Acrylamidyl Cystenyl	174
N-Glucuronyl (N terminus)	176
delta-Glycosyloxy- (of Lysine) or beta-Glycosyloxy- (of Phenylalanine or Tyrosine)	177
4-Glycosyloxy- (hexosyl,C6) (of Proline)	177
Benzyl Seryl	177
N-methyl Tyrosinyl	177
a-N-Gluconoylation (His Tagged proteins)	178
p-Nitrobenzyloxycarbonyl (4Nz)	179
2,4,5-Trichlorophenyl	179
2,4,6-trimethyloxybenzyl (Tmob)	180
Xanthyl (Xan)	180
Tyrosine (Y, Tyr)	181
Chlorophenylalanyl	182
Mesitylene-2-sulfonyl (Mts)	182
AEBSF	183
Isopropyl Lysyl	184
Tryptophanyl (-W-, - Trp-)	186
Carboxymethyl Lysyl	186
N-Lipoyl- (on Lysine)	188
Matrix alpha cyano MH+	190
Benzyl Threonyl	191
Benzyl Cystenyl	193
Napthyl Alanyl	197
Succinyl Aspartamyl	198
HMP (hydroxymethylphenyl)/TFA adduct	201
N-acetylhexosamines (GalNAc, GlcNAc)	203
Tryptophan (W, Trp)	204
Cystine ((Cys)2)	204
Farnesylation	204
S-Farnesyl-	206
Myristoylation-4H (2 double bonds)	206
Myristoleylation (myristoyl with one double bond)	208
Pyridylethyl Cystenyl	208
Myristoylation	210
4-Methoxy-2,3,6-trimethylbenzenesulfonyl (Mtr)	212
2-Bromobenzyloxycarbonyl (BrZ)	213
Formyl Tryptophanyl	214
Benzyl Glutamyl	219
Anisole Adducted Glutamyl	219
9-Fluorenylmethyloxycarbonyl (Fmoc)	222
S-cystenyl Cystenyl	222
Biotinylation (amide bond to lysine)	226
Dimethoxybenzhydryl (Mbh)	226
N-Pyridoxyl (on Lysine)	229
Pyridoxal phosphate (Schiff Base formed to lysine)	231
Dansyl (Dns)	233
Nicotinyl Lysyl	233

Modyfikacja	Zmiana masy cząsteczki [Da]
2-(p-biphenyl)isopropyl-oxycarbonyl (Bpoc)	238
Palmitoylation	238
Triphenylmethyl (Trityl, Trt)	242
Tyrosinyl Sulphate	243
Phospho Tyrosinyl	243
Pbf (pentamethyldihydrobenzofuransulfonyl)	252
3,5-Diiodination (of Tyrosine)	252
a-N-6-Phosphogluconoylation (His Tagged proteins)	258
N alpha -(gamma-Glutamyl)-Glu2	259
O-GlcNAc-1-phosphorylation (of Serine)	266
Stearoylation	266
Pmc (2,2,5,7,8- Pentamethylchroman-6-sulphonyl)	266
Geranylgeranylation	272
Monomethoxytrityl	272
S-Geranylgeranyl	276
5'phos dCytidinyl	289
iodo Tyrosinyl	289
Aldohexosyl Lysyl	290
Sialyl	291
N-acetylneuraminic acid (Sialic acid, NeuAc, NANA, SA)	291
5'phos dThymidinyl	304
5'phos Cytidinyl	305
Glutathionation	305
O-Uridinylylation (of Tyrosine)	306
5'phos Uridinyl	306
N-glycolneuraminic acid (NeuGc)	307
S-farnesyl Cystenyl	307
5'phos dAdenosyl	313
O-pantetheinephosphorylation (of Serine)	324
SucPhencarb Lysyl	327
5'phos dGuanosyl	329
5'phos Adenosinyl	329
O-5'-Adenosylation (of Tyrosine)	329
4'-Phosphopantetheine	339
S-palmityl Cystenyl	342
5'phos Guanosyl	345
Biotinyl Lysyl	354
Fluorescein labelling of peptide N-terminal using NHS ester	359
Hex-HexNAc	365
N alpha -(gamma-Glutamyl)-Glu3	388
Dioctyl Phthalate	391
PMC Lysyl	395
Aedans Cystenyl	409
Dioctyl Phthalate Sodium Adduct	413
di-iodo Tyrosinyl	415
PMC Arginyl	423
S-Coenzyme A	454
AMP Lysyl	457

Modyfikacja	Zmiana masy cząsteczki [Da]
3,5,3'-Triiodothyronine (from Tyrosine)	470
S-(sn-1-Dipalmitoyl-glyceryl)- (on Cysteine)	524
S-(ADP-ribosyl)- (on Cysteine)	541
N-(ADP-ribosyl)- (on Arginine)	541
O-ADP-ribosylation (on Glutamate or C terminus)	541
ADP-rybosylation (from NAD)	541
S-Phycocyanobilin (on Cysteine)	587
S-Heme (on Cysteine)	617
N theta -(ADP-ribosyl) diphthamide (of Histidine)	648
NeuAc-Hex-HexNAc	657
O-8 alpha-Flavin [FAD])- (of Tyrosine)	783
S-(6-Flavin [FAD])- (on Cysteine)	784
N theta and N pi-(8alpha-Flavin) (on Histidine)	784
(Hex)3-HexNAc-HexNAc	893
(Hex)3-HexNAc-(dHex)HexNAc	1039

A.3 Lista aminokwasów

Tabela A.2: Lista aminokwasów wraz z ich masą cząsteczkową i 1-literowym skrótem FASTA

Aminokwas	Kod FASTA	Średnia masa cząsteczkowa [Da]
Alanina	A	71.03711
Arginina	R	156.10111
Asparagina	N	114.04293
Cysteina	C	103.00919
Fenyloalanina	F	147.06841
Glutamina	Q	128.05858
Glicyna	G	57.02146
Histydyna	H	137.0589
Izoleucyna	I	113.08406
Kwas asparaginowy	D	115.02694
Kwas glutaminowy	E	129.04259
Leucyna	L	113.08406
Lizyna	K	128.09496
Metionina	M	131.04049
Prolina	P	97.05276
Seryna	S	87.03203
Treonina	T	101.04768
Tryptofan	W	186.07931
Tyrozyna	Y	163.06333
Walina	V	99.06841

Bibliografia

[1] The structures of life. National Institute of General Medical Sciences, 2008.

[2] J. S. A. Kraj. Proteomika. EJB, Kraków, 2004.

[3] L. M. Adleman. Phylogenetic graph models beyond trees. *Science*, 5187(266):1021–1024, 1994.

[4] J. Arabas. Wykłady z algorytmów ewolucyjnych. WNT, Warszawa, 2001.

[5] J. Błażewicz. Złożoność obliczeniowa problemów kombinatorycznych. WNT, Warszawa, 1988.

[6] J. Błażewicz, M. Borowski, P. Formanowicz, and T. Głowacki. On graph theoretical models for peptide sequence assembly. Foundations of Computing and Decision Sciences, 30:183–191, 2005.

[7] J. Błażewicz, M. Borowski, P. Formanowicz, and T. Głowacki. Genetic and tabu search algorithms for peptide assembly problem. RAIRO - Operations Research, 44:153–166, 2010.

[8] J. Błażewicz, A. Hertz, D. Kobler, and D. de Werra. On some properties of DNA graphs. Discrete Applied Mathematics, 98:1–19, 1999.

[9] J. Błażewicz, A. Hertz, D. Kobler, and D. de Werra. Phylogenetic graph models beyond trees. Discrete Applied Mathematics, 157:2361–2369, 2009.

[10] J. Błażewiczwicz, M. Borowski, P. Formanowicz, and M. Stobiecki. Tabu search method for determining sequences of amino acids in long polypeptides. Lecture Notes in Computer Science, 3449:22–32, 2005.

[11] R. Bellman. On the theory of dynamic programming. Proceedings of the National Academy of Sciences of the United States of America, 38:716–719, 1952.

[12] C. Berge. Graphes. Dunod, Paris, 1970.

[13] J. Blazewicz. Złożoność obliczeniowa problemów kombinatorycznych. Wydawnictwa Naukowo-Techniczne, Warszawa, 1988.

[14] J. Blazewicz, M. Kasprzak, B. Leroy-Beaulieu, and D. de Werra. Finding hamiltonian circuits in quasi-adjoint graphs. Discrete Applied Mathematics, 156(13):2573 – 2580, 2008.

[15] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, and A. J. Smola. Protein function prediction via graph kernels. Bioinformatics (Suppl 1), 21:47–56, 2005.

[16] B. M. C. Xu. Complexity and scoring function of ms/ms peptide de novo sequencing. Computational Systems Bioinformatics / Life Sciences Society. Computational Systems Bioinformatics Conference, pages 361–369, 2006.

[17] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. On some properties of dna graphs. Discrete Applied Mathematics, 98:1–19, 2001.

[18] R. Craig and R. Beavis. A method for reducing the time required to match protein sequences with tandem mass spectra. Rapid Commununications in mass spectrometry: RCM, pages 2310–2316, 2003.

[19] F. Crick. Central dogma of molecular biology. Nature, page 561–563, 1970.

[20] J. R. Y. D. L. Tabb, A. Saraf. Gutentag: High-throughput sequence tagging via an empirically derived fragmentation model. Analytical Chemistry, 75:6415–6421, 2003.

[21] N. G. de Bruijn. A combinatorial problem. Koninklijke Nederlandse Akademie v. Wetenschappen, 49:758–764, 1946.

[22] E. W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1:269–271, 1959.

[23] S. Doonan. Peptides and Proteins. Royal Society of Chemistry, Exeter, 2002.

[24] J. K. Eng, A. L. McCormack, and J. R. Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. Journal of the American Society for Mass Spectrometry, 5:976–989, 1994.

[25] L. Euler. Solutio problematis ad geometriam situs pertinentis. Graph Theory 1736-1936, February 1736.

[26] P. Formanowicz. Selected combinatorial aspects of biological sequence analysis. Wydawnictwo Politechniki Poznańskiej, 2005.

[27] J. K. Gallant. The complexity of the overlap method for sequencing biopolymers. Journal of Theoretical Biology, 101:1–17, 1983.

[28] T. Głowacki, A. Kozak, M. Borowski, and P. Formanowicz. O algorytmach asemblacji długich łańcuchów peptydowych. Automatyzacja procesów dyskretnych, Teoria i zastosowania, II:45–52, 2010.

[29] T. Głowacki, A. Kozak, P. Rek, and P. Formanowicz. Metody asemblacji długich łańcuchów peptydowych i ich złożoność obliczeniowa. 2012.

[30] E. D. Hoffmann, J. J. Charette, V. Stroobant, and L. Konopski. Spektrometria Mas. WNT, 1998.

[31] N. M. Hooper. Proteases in Biology And Medicine (Essays in Biochemistry). Portland Press Ltd, 2002.

[32] R. A. W. Johnstone. Mass spectrometry for organic chemists. Cambridge University Press, Cambridge, 1972.

[33] D. E. Knuth. Art of Computer Programming, Volume 1: Fundamental Algorithms. Addison-Wesley Professional, wydanie 3, Massachusetts, 1997.

[34] A. Kozak, T. Głowacki, and P. Formanowicz. Constructing oligonucleotide libraries based on graph theory models. Ifl Technical report Series, 06:39–42, 2008.

[35] A. Kozak, T. Głowacki, and P. Formanowicz. Klasyfikacja problemów asemblacji i sekwencjonowania peptydów. 2012.

[36] S. Luke. Essentials of metaheuristics. http://cs.gmu.edu/∼sean/book/metaheuristics/, 2009.

[37] D. J. M.R. Garey. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979.

[38] C. H. Papadimitriou. Złożoność obliczeniowa. Wydawnictwa Naukowo-Techniczne, Warszawa, 2002.

[39] D. N. Perkins, D. J. Pappin, D. M. Creasy, and J. S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. Electrophoresis, 20:3551–3567, 1994.

[40] T. N. R. Westermeier. Proteomics in Practice: A Laboratory Manual of Proteome Analysis. Wiley-VCH, Weinheim, 2002.

[41] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. Handbook of Metaheuristics, pages 716–719, 2003.

[42] L. A. Ross and C. R. Wright. Discrete Mathematics (5th edition). Prentice Hall, 1988.

[43] C. D. W. S. B. Needleman. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48:443–453, 1970.

[44] B. C. Searle, S. Dasari, M. Turner, A. P. Reddy, D. Choi, P. A. Wilmarth, A. L. McCormack, L. L. David, and S. R. Nagalla. High-throughput identification of proteins and unanticipated sequence modifications using a mass-based alignment algorithm for ms/ms de novo sequencing results. Analytical Chemistry, 76:2220–2230, 2004.

[45] L. Stryer. Biochemistry (4th edition). W.H. Freeman & Company, New York, 1995.

[46] P. F. T. Głowacki, A. Kozak. Asemblacja długich łańcuchów peptydowych przy wykorzystaniu metaheurystyki grasp. Zeszyty Naukowe Politechniki Śląskiej, 150:203–209, 2008.

[47] T.Chen, M. Kao, M.Tepel, J.Rush, and G. Church. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. Journal of Computational Biology, pages 325–337, 2001.

[48] A. Uzawa and T. Oshima. Polypeptide synthesis directed by dna as a messenger in cell-free polypeptide synthesis by extreme thermophiles. The Journal of Biochemistry, 131:849–853, 2002.

[49] K. Z. Y. Han, B. Ma. Spider: Software for protein identification from sequence tags with de novo sequencing error. Journal of Bioinformatics and Computational Biology, 3:697–716, 2005.

[50] S.-Y. Ying. Complementary DNA libraries. Methods in Molecular Biology, 221, February 2003.