POLITECHNIKA POZNAŃSKA

Wydział Informatyki
Instytut Informatyki

Kombinatoryczne aspekty nieklasycznego sekwencjonowania DNA przez hybrydyzację

Marcin Radom

Rozprawa doktorska

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

Poznań 2011

Serdecznie dziękuję

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

najbliższym za wsparcie

Spis treści

Spis rysunków

Spis tablic

Skróty i oznaczenia

DNA	Kwas deoksyrybonukleinowy (ang. **D**eoxy **R**ybonucleine **A**cid)
RNA	Kwas rybonukleinowy (ang. **R**ybo**N**ucleine **A**cid)
SBH	Sekwencjonowanie przez hybrydyzację (ang. **S**equencing **B**y **H**ybridization)
ASBH	Sekwencjonowanie przez hybrydyzację w wersji Alternating (ang. **A**lternating **SBH**)
BSBH	Sekwencjonowanie przez hybrydyzację w wersji Binary (ang. **B**inary **SBH**)
GSBH	Sekwencjonowanie przez hybrydyzację w wersji Gapped (ang. **G**apped **SBH**)

Rozdział 1

Wprowadzenie

1.1 Uzasadnienie tematu badawczego

Bardzo trudno jest wyznaczyć precyzyjne daty, o których można bez żadnych wątpliwości stwierdzić, że rzeczywiście zapoczątkowały którąś z nauk rozwijanych we współczesnym świecie. Łatwiejszym jest podanie momentów odkryć czy publikacji, które okazały się kamieniami milowymi dla rozwoju wielu gałęzi nauki. Taką przełomową datą dla biologii molekularnej było na pewno pojawienie się publikacji dotyczącej odkrycia struktury kwasu deoksyrybonukleinowego przez Jamesa Watsona i Francisa Cricka w czasopiśmie *Nature* w 1953 [53]. Dzięki tej i kolejnym publikacjom rozwinęły się zupełnie nowe gałęzie nauki wywodzące się zarówno z biologii jak i z matematyki oraz nieco później z informatyki. Biologia molekularna, biologia obliczeniowa, bioinformatyka oraz całe spektrum specjalizacji stanowią jasny dowód na rozwój tych dziedzin nauki, dla których odkrycie z 1953 było jednym z kluczowych.

Odkrycie struktury kwasu deoksyrybonukleinowego, oznaczanego o wiele lepiej rozpoznawalnym skrótem - DNA, umożliwiło znaczne poszerzenie możliwości w sferze badań nad organizmami żywymi. Centralny dogmat biologii molekularnej [13, 14], o którym będzie szerzej mowa w następnym rozdziale dotyczącym aspektów biologicznych niniejszej rozprawy, mówiąc w uproszczeniu pokazuje przejście informacji od DNA do białek, będących głównymi 'cegiełkami' budującymi wszystkie znane żywe organizmy. Samo DNA jest w biologii, w zależności od potrzeby, dzielone na mniejsze jednostki jak regiony, chromosomy, geny, sekwencje kodujące, DNA 'śmieciowe' (*ang. junk DNA*) itd.

Jakkolwiek zaawansowane i nowatorskie są jednak dziedziny nauki związane z analizą funkcji genów, mapowaniem genomów, genomiką porównawczą, strukturami wyższych rzędów jak białka i ich modele przestrzenne, a nawet wzajemne interakcje całych grup genów oddziałujących w żywych organizmach, u podstaw tych oraz wielu innych pól

zainteresowań biologii leży poznanie sekwencji DNA. Odkrycie DNA otwarło więc nowy problem badawczy jakim stały się próby odczytania sekwencji DNA pochodzącej z tychże organizmów.

Proces odczytu DNA nosi nazwę sekwencjonowania, czyli innymi słowy precyzyjnego określenia sekwencji czterech rodzajów nukleotydów budujących kwas deoksyrybonukleinowy. Dopiero od momentu poznania tejże sekwencji można podejmować inne badania, których celem jest zrozumienie informacji zapisanej w DNA. Jest zatem oczywiste, że sukces w tym pierwszym i elementarnym kroku musi zostać osiągnięty, aby możliwe stało się prowadzenie dalszych badań.

Istnieje wiele różnych metod sekwencjonowania, poczynając od historycznych, nie używanych już w szerszym zakresie badań, po najnowsze, za pomocą których można szybko i tanio odczytać łańcuchy DNA o długościach niewyobrażalnych w pionierskich latach odkryć dotyczących struktur DNA. Wszystkie metody mają swoje wady i zalety, podlegają nieustannym modyfikacjom i ulepszeniom, a wiele z nich jest nadal stosowanych pomimo lat, które minęły od ich opisania w literaturze naukowej. Można powiedzieć, że wiele z nich znajduje swoją niszę zastosowań, w której przy określonych warunkach stanowią dobrą alternatywę nawet wobec najnowszych rozwiązań w dziedzinie sekwencjonowania.

Metodą, która stanowi główny przedmiot niniejszej rozprawy doktorskiej jest sekwencjonowanie przez hybrydyzację (ang. Sequencing by Hybrydization, SBH). Metoda ta wykorzystuje jedną z podstawowych cech DNA - komplementarność. Jest to reguła, w myśl której każdy z czterech rodzajów nukleotydów budujących DNA ma swój precyzyjnie określony odpowiednik w drugiej nici z dwóch budujących tzw. podwójną helisę DNA. W wyniku klasycznego eksperymentu hybrydyzacyjnego sklonowane, jednoniciowe DNA łączy się z chipem DNA. Na jego powierzchni znajdują się wszystkie sekwencje oligonukleotydów o zadanej długości, w wyniku czego teoretycznie możliwa jest do uzyskania informacja o wszystkich podłańcuchach o danej długości tworzących badaną cząsteczkę DNA. W myśl teorii, w przypadku idealnym informacja ta powinna być kompletna. Wykorzystując algorytmy kombinatoryczne możliwe staje się odtworzenie precyzyjnej i dokładnej struktury badanego DNA, co stanowi właśnie główny cel sekwencjonowania. Jak łatwo zauważyć metoda sekwencjonowania przez hybrydyzację składa się z dwóch faz – biochemicznej oraz obliczeniowej. Eksperyment biochemiczny (hybrydyzacyjny) niezbędny jest do uzyskania zbioru podciągów badanego DNA. Zbiór ten będzie dalej w pracy nazywany *spektrum DNA*. Stanowi ono dane wejściowe dla fazy obliczeniowej. Od tego momentu wymagane jest użycie metod kombinatorycznych, aby odtworzyć kolejność nukleotydów w badanej cząsteczce DNA.

Jak widać z powyższego opisu, metoda sekwencjonowania przez hybrydyzację łączy w sobie zagadnienia biologiczne oraz informatyczne. To połączenie występujące także

w wielu innych problemach badawczych, w ramach szeroko pojętej biologii molekularnej, stanowi jeden z powodów nazywania nauki zajmującej się rozwiązywaniem takich zagadnień mianem biologii obliczeniowej. Biologia obliczeniowa jest interdyscyplinarną gałęzią nauki łączącą zarówno biologię i informatykę, jak również statystykę, czy w ogólności matematykę w celu rozwiązywania problemów biologicznych o wysokim poziomie złożoności.

Jak już wspomniano, każda metoda sekwencjonowania ma wady i zalety. Współczesne, szybkie i nowoczesne sekwenatory DNA bazujące na przykład na metodzie pirosekwencjonowania (jak metoda potocznie zwana sekwencjonowaniem 454 [34]), również nie rozwiązują wszystkich problemów. W omawianym przypadku problem pojawia się w momencie wystąpienia dużej liczby powtórzeń tych samych nukleotydów w pewnym fragmencie DNA, co może prowadzić do braku precyzji w określeniu ich liczby. Zaproponowane przez naukowców metody sekwencjonowania DNA są bardzo często modyfikowane. Nie inaczej jest z SBH. Metoda sekwencjonowania przez hybrydyzację liczy sobie już ponad 20 lat, jednak nadal jest intensywnie badana i rozwijana, co ma odzwierciedlenie w liczbie nowych publikacji na jej temat pojawiających się każdego roku w literaturze naukowej. Od końca lat 80-tych, kiedy została opublikowana doczekała się ona bardzo licznych ulepszeń, jak użycie jej w ramach następujących po sobie rund hybrydyzacji [22, 33, 49] czy używania dodatkowej informacji poza samym spektrum DNA, jak na przykład przybliżone położenie oligonukleotydów w całej sekwencji [2, 24, 56]. Warto także wymienić takie podejścia jak użycie izotermicznych bibliotek oligonukleotydów na powierzchni chipu DNA [5, 6, 8] czy znajomość informacji o powtórzeniach [21, 28]. Wiele różnych metod sekwencjonowania DNA, w tym metoda SBH zostanie dokładniej opisanych w kolejnych rozdziałach niniejszej rozprawy.

Podsumowując, można powiedzieć, że metoda SBH jest precyzyjnym sposobem określenia sekwencji DNA, jednakże jest ograniczona długością badanych sekwencji jak i pojemnością chipu, a więc technologią wytwarzania mikromacierzy DNA w niej wykorzystywanych. Niniejsza rozprawa doktorska koncentruje się na jednym ze sposobów usprawnienia tejże metody poprzez wykorzystanie nieklasycznych chipów DNA w fazie biochemicznej SBH. Pociąga to za sobą konieczność sformułowania i analizy odpowiednich problemów kombinatorycznych, które należy rozważyć w fazie obliczeniowej, oraz konstrukcji algorytmów rozwiązujących te problemy.

1.2 Cel i zakres pracy

Bazując na powyższej wiedzy można określić założenia dla dalszej pracy, związane z cechami charakterystycznymi metody sekwencjonowania przez hybrydyzację:

1. Sekwencjonowanie przez hybrydyzację umożliwia dokładne i precyzyjne odczytywanie długich sekwencji DNA. Należy podkreślić potrzebę precyzji, zwłaszcza w kontekście części obliczeniowej SBH. Często pojawia się tutaj problem tzw. rozwiązań niejednoznacznych, to znaczy wielu równorzędnych sekwencji DNA spełniających wszystkie warunki stawiane dla rozwiązań rozpatrywanego problemu obliczeniowego. Stanowi to istotną wadę, ponieważ nie można bez dodatkowych badań ustalić, którą z otrzymanych sekwencji DNA jest ta badana w danym eksperymencie hybrydyzacyjnym.

2. Głównym ograniczeniem efektywności są błędy hybrydyzacji oraz złożoność obliczeniowa podejścia klasycznego. W przypadku braku błędów hybrydyzacji problem ten jest prosty [39]. Jeśli jednak występują one w spektrum, to problem sekwencjonowania przez hybrydyzację staje się trudny z punktu widzenia złożoności obliczeniowej [10]. W swojej klasycznej formie problem sekwencjonowania przez hybrydyzację z błędami należy do klasy problemów silnie NP-trudnych, tj. nie istnieje dla niego dokładny algorytm wielomianowy, o ile $P \neq NP$. Ograniczanie długości badanej sekwencji ma na celu zmniejszenie liczby błędów hybrydyzacji, a co za tym idzie minimalizację liczby rozwiązań niejednoznacznych.

3. Błędy hybrydyzacji wpływają na liczbę rozwiązań niejednoznacznych, czyli różnych rekonstrukcji badanego DNA przy braku wiedzy, która z otrzymanych sekwencji jest prawdziwym analizowanym DNA.

4. Szybkie, precyzyjne i jednoznaczne sekwencjonowanie łańcuchów DNA umożliwia dalsze badania nad żywymi organizmami, przez co stanowi w pewien sposób ich podstawę. Aby zapewnić wysoką jakość takich badań należy dołożyć starań, aby otrzymane za pomocą SBH sekwencje DNA były pozbawione błędów wynikających ze wspomnianego już problemu rozwiązań niejednoznacznych.

Celem niniejszej pracy jest analiza problemu sekwencjonowania DNA przez hybrydyzację w wersji nieklasycznej, czyli dość odmiennej od podstawowej wersji metody SBH zaproponowanej w latach 1989-1990. Praca niniejsza bazuje między innymi na zaproponowanej w 1994 roku koncepcji zastosowania jednego z trzech rodzajów nieklasycznych chipów DNA [40]. Chipy takie zawierają sondy, które w przeciwieństwie do podejścia klasycznego składają się nie z jednego konkretnego typu oligonukleotydu, lecz z pewnego ich zbioru. Spektra DNA uzyskane w eksperymencie hybrydyzacyjnym przy użyciu tego rodzaju chipów znacznie różnią się od tych uzyskanych we wcześniej wspomnianym podejściu klasycznym. W spektrum klasycznym mamy do czynienia z podciągami o danej długości z sekwencji DNA, w rozpatrywanym podejściu nieklasycznym element spektrum reprezentuje pewien zbiór podciągów. Z uzyskanych danych możliwe jest odtworzenie

badanego DNA, lecz do tego celu wymagane są zupełnie nowe algorytmy rekonstrukcji, dedykowane dla każdego chipu osobno. Jednakże nawet w najprostszej wersji algorytmy stworzone w ramach poprzednich prac [45] wykazują o wiele lepszą skuteczność niż czyste podejście klasyczne. Główne cele i zadania niniejszej pracy doktorskiej są określone następująco:

1. Opracowanie precyzyjnych matematycznych definicji problemów nieklasycznego sekwencjonowania przez hybrydyzację oraz określenie ich złożoności obliczeniowej.

2. Opracowanie dowodów złożoności rozpatrywanych problemów sekwencjonowania nieklasycznego. Problemy kombinatoryczne badane są zarówno w podstawowej wersji z wyłączeniem błędów hybrydyzacji jak i z uwzględnieniem błędów typu negatywnego i pozytywnego, które wpływają na precyzję danych zawartych w spektrum DNA. Klasyczne podejście do SBH z uwzględnieniem błędów hybrydyzacji zostało jednoznacznie określone jako problem silnie NP-trudny. Przynależność do klas złożoności rozpatrywanych tutaj problemów w wersji z błędami nie została do tej pory przebadana.

3. Stworzone zostały już algorytmy dla trzech nieklasycznych chipów DNA radzące sobie tylko z występowaniem w spektrum błędów negatywnych wynikających z powtórzeń [45]. Kolejnym celem pracy jest więc opracowanie nowych algorytmów zdolnych rekonstruować DNA w warunkach występowania błędów pozytywnych i negatywnych hybrydyzacji. Jest to uzasadnione realistycznym założeniem, że nie jest w pełni możliwe uniknięcie błędów hybrydyzacji podczas fazy biochemicznej SBH. Praktycznie przydatne algorytmy muszą więc uwzględniać ich występowanie oraz radzić sobie z nimi w najlepszy możliwy sposób. Podejście klasyczne przy uwzględnieniu błędów hybrydyzacji narażone jest na występowanie licznych rozwiązań niejednoznacznych, tak więc jednym z celów opracowania algorytmów dla podejścia nieklasycznego jest redukowanie liczby takich rozwiązań w czasie rekonstrukcji DNA z informacji ze spektrum obarczonego błędami hybrydyzacji.

4. Celem opracowywanych algorytmów jest nie tylko sekwencjonowanie DNA przy użyciu nieklasycznego podejścia do SBH, ale także uzyskanie jak największej precyzji otrzymanych rozwiązań poprzez jak najefektywniejsze wykorzystanie informacji ze spektrum. Innymi słowy, celem jest więc otrzymywanie rozwiązań jednoznacznych dla badanego DNA oraz określenie optymalnych parametrów chipów nieklasycznych, dla DNA o zadanym zakresie długości.

5. Opracowanie precyzyjnych testów i porównań trzech rozpatrywanych chipów DNA ze sobą oraz z wynikami uzyskanymi przy użyciu podejścia klasycznego do SBH. Wiedza płynąca z takich testów stanowiłaby istotny wkład w ułatwienie wyboru

najlepszego podejścia, w zależności od problemu sekwencjonowania DNA, z którym w danej chwili należy się zmierzyć.

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

W rozdziale I przedstawiono problem podejścia sekwencjonowania przez hybrydyzację oraz główne założenia przeprowadzonych badań opisanych w niniejszej pracy.

W rozdziale II przedstawione zostaną podstawowe zagadnienia informatyki teoretycznej niezbędne do pełnego zrozumienia zawartości dalszych rozdziałów, w szczególności części poświęconych złożoności obliczeniowej problemów kombinatorycznych.

W rozdziale III opisane zostały podstawy chemii bioorganicznej, w szczególności dokładny i precyzyjny opis budowy DNA.

Rozdział IV pracy dotyczy opisu ważniejszych metod sekwencjonowania, zarówno najwcześniejszych jak i współczesnych, opartych na sekwenatorach DNA. Szczególną uwagę poświęcono zagadnieniu sekwencjonowania przez hybrydyzację wraz z opisem różnych podejść stosowanych od czasu opublikowania całej idei, będącej dalej w pracy nazywanej podejściem klasycznym.

W rozdziałach V, VI oraz VII zostały przedstawione trzy chipy nieklasyczne: Gapped Chip, Alternating Chip, Binary Chip, zagadnienia złożoności obliczeniowej oraz algorytmy dla nich. Podstawowa struktura tych rozdziałów jest taka sama dla każdego chipu. Na wstępie opisana jest budowa danego chipu oraz jego cechy charakterystyczne. Następnie rozdział zawiera sformułowania problemów kombinatorycznych z uwzględnieniem występowania błędów hybrydyzacji. Po formalnych definicjach problemów następuje część rozdziału dotycząca złożoności problemów kombinatorycznych. Ostatni podrozdział zawiera dokładny opis konstrukcji i działania algorytmu opracowanego dla danego chipu.

W rozdziale VIII na wstępie przedstawione zostaną główne założenia testów dla opracowanych algorytmów, sposoby porównywania wyników oraz źródła sekwencji testowych. Następnie rozdział ten zawiera dokładne wyniki testów przeprowadzonych dla algorytmów oraz wnioski z nich wypływające.

Rozdział IX stanowi zakończenie niniejszej rozprawy doktorskiej, w formie szerokiego podsumowania osiągniętych wyników. Podsumowano w nim opracowane algorytmy i dowody złożoności problemów z użyciem chipów nieklasycznych. Znalazło się w nim również zwięzłe podsumowanie przeprowadzonych testów dla odpowiednich algorytmów oraz otwierające się możliwości przyszłych prac badawczych.

Rozdział 2

Zagadnienia złożoności obliczeniowej, teorii algorytmów oraz teorii grafów

2.1 Alfabet, słowa, języki

W tym i w dalszych rozdziałach niniejszej rozprawy będzie mowa o ciągach znaków oraz operacjach na nich wykonywanych. Zasadne jest więc określenie odpowiednich definicji najbardziej podstawowych pojęć.

Def. 2.1.1. Alfabet jest skończonym zbiorem symboli lub znaków, oznaczanym dalej przez Σ.

Przykładem może być alfabet binarny oznaczony jako $\Sigma = \{0, 1\}$. Przez Σ^* oznaczać będziemy zbiór wszystkich skończonych ciągów dających się wyrazić za pomocą alfabetu Σ.

Def. 2.1.2. Słowo zbudowane nad alfabetem Σ jest to uporządkowany zbiór znaków tego alfabetu mający pewną długość. Słowa będą dalej zamiennie określane mianem *ciągów*, lub *ciągów znaków*.

Ciągami zbudowanymi nad przykładowym alfabetem binarnym (tj. słowami zbudowanymi *nad* tym alfabetem) będą więc 0, 01, 10, 1101101, itd. Jeżeli przez s oznaczymy pewien ciąg, wtedy $|s|$ oznaczać będzie jego długość, przykładowo $s = 00101$, $|s| = 5$.

Mając dane ciągi s oraz z, przez $s \cdot z$ oznaczać będziemy konkatenację tych ciągów, czyli nowy ciąg t o długości $|t| = |s| + |z|$ składający się z kolejno występujących liter ciągu s a następnie kolejno występujących liter ciągu z.

Def. 2.1.3. Dwa ciągi s i z nakładają się na siebie na k znakach, jeżeli istnieją takie ciągi a, b, c, że $|b| = k$ oraz $s = a \cdot b$ i $z = b \cdot c$.

W podanej definicji ciąg b jest wspólnym podciągiem ciągów s oraz z, czyli jak dalej będzie to określane: ich wspólnym *nałożeniem*.

2.2 Teoria grafów

Ponieważ w pracy pojawiać się będą zagadnienia dotyczące grafów, zarówno w sekcjach dotyczących złożoności omawianych problemów jak i w częściach poświęconych opracowanym algorytmom, zasadne jest choć krótkie wprowadzenie ważniejszych pojęć i definicji związanych z teorią grafów.

Graf opisujemy poprzez zdefiniowanie zbioru wierzchołków oraz zbioru krawędzi / łuków.

Def. 2.2.1. Graf nieskierowany G jest uporządkowaną parą zbiorów $G = (V, E)$, gdzie V oznacza zbiór wierzchołków, E zbiór krawędzi.

Zbiór E jest więc podzbiorem zbioru wszystkich par $\{\{v, w\} : v, w \in V\}$.

Def. 2.2.2. Graf skierowany G jest uporządkowaną parą zbiorów $G = (V, A)$, gdzie V oznacza zbiór wierzchołków, A zbiór łuków.

Zbiór A jest więc podzbiorem zbioru wszystkich par $\{\{v, w\} : v, w \in V\}$. Mówimy, że z wierzchołka v do wierzchołka w prowadzi łuk, jeżeli para (v, w) należy do zbioru A. Z wierzchołka w do wierzchołka v prowadzi łuk, jeżeli $(w, v) \in A$.

Ścieżka w grafie to uporządkowany zbiór wierzchołków ze zbioru V taki, że każda kolejna para wierzchołków tego zbioru należy do zbioru krawędzi E lub łuków A grafu oraz wszystkie kolejne pary mają wspólny odpowiednio ostatni i pierwszy wierzchołek należący do V. Na przykład dwie uporządkowane pary krawędzi (v, w) i (w, z) są fragmentem ścieżki w grafie $G = (V, E)$ jeżeli $v, w, z \in V : (v, w), (w, z) \in E$.

Cyklem w grafie nazywać będziemy taką ścieżkę, w której wierzchołek początkowy jest równy wierzchołkowi końcowemu ścieżki. Dwie ścieżki / cykle w teorii grafów o których będzie mowa w dalszych rozdziałach to ścieżka/cykl Eulera oraz ścieżka/cykl Hamiltona.

Def. 2.2.3. Ścieżka Eulera w grafie $G = (V, E)$ to taka ścieżka, która przechodzi przez każdą krawędź $e \in E$ dokładnie raz.

Def. 2.2.4. Ścieżka Hamiltona w grafie $G = (V, E)$ to taka ścieżka, która przechodzi przez każdy wierzchołek $v \in V$ dokładnie raz.

Odpowiednio cykle Eulera i Hamiltona to takie ścieżki, w których wierzchołek początkowy s jest równy wierzchołkowi końcowemu t ścieżki.

Grafy można podzielić na bardzo wiele kategorii. W dalszych rozważaniach rozprawy zakładane będzie istnienie w odpowiednich instancjach problemów tzw. grafów spójnych.

Def. 2.2.5. Graf jest określany jako spójny, jeżeli między dowolną parą wierzchołków $v, w \in V$ istnieje ścieżka.

Liczność zbiorów wierzchołków oraz krawędzi będziemy dalej określać odpowiednio przez $|V|$ oraz $|E|$. Przydatnym okaże się też pojęcie *stopnia* wierzchołka grafu. Stopniem wierzchołka v oznaczanym jako δv nazywamy liczbę krawędzi z nim sąsiadujących, tj. takich, które występują w zbiorze krawędzi E grafu, mając ten wierzchołek na pierwszym lub drugim miejscu w parze określającej krawędź.

W przypadku grafu skierowanego $G = (V, A)$ stopień może określać zarówno łuki przychodzące, jak i wychodzące z wierzchołka. W tym wypadku mówić możemy też o liczbie następników i poprzedników wierzchołka. Liczba następników wierzchołka v to liczba wierzchołków, do których w zbiorze łuków A istnieją łuki wychodzące z wierzchołka v. Analogicznie liczba poprzedników wierzchołka v to liczba wierzchołków, z których istnieją łuki w zbiorze A prowadzące bezpośrednio do v.

2.3 Podstawowe zagadnienia złożoności obliczeniowej

Problem sekwencjonowania przez hybrydyzację należy przedstawić w formalny, matematyczny sposób, aby możliwa była analiza jego złożoności oraz zaprojektowanie rozwiązujących go algorytmów. Zanim w kolejnych rozdziałach pracy zostaną wprowadzone definicje oraz odmiany rozpatrywanych problemów sekwencjonowania w oparciu o podejście nieklasyczne, w niniejszym rozdziale przedstawiona zostanie niezbędna teoria stojąca za dalszymi badaniami.

Def. 2.3.1. Przez *problem* Π oznaczać dalej będziemy uporządkowaną parę ciągów typu (I, A), gdzie I to *instancja* problemu Π, A stanowi zaś odpowiedź dla tej instancji.

Problem określony zostaje poprzez zbiór jego parametrów lub zmiennych składających się nań, oraz co równie ważne - poprzez zdefiniowanie wszystkich warunków jakie musi spełniać jego rozwiązanie. *Instancja* I problemu powstaje poprzez precyzyjne określenie wszystkich wartości parametrów dla problemu. Dla przykładu problemem jest tzw. problem komiwojażera (ang. Travelling Salesman Problem, TSP), który określa się poprzez listę miast oraz odległości pomiędzy każdą ich parą. Rozwiązaniem problemu komiwojażera jest uszeregowana lista miast o pewnym koszcie przejścia (tutaj: sumie odległości). Konkretna instancja TSP zawiera więc określony zbiór miast C, tabelę z wartościami odległości pomiędzy miastami w formie: $d(c_x, c_y) = w$ oznaczającą odległość z miasta c_x do c_y równą w. Dana jest pewna wartość W_{max} określająca rozwiązanie dopuszczalne, czyli takie, w którym uporządkowana lista miast, z którą związany jest pewien sumaryczny koszt przejścia w_{sol} od pierwszego do ostatniego miasta na liście, spełnia warunek $w_{sol} \leqslant W_{max}$. W problemie przyjmowany jest warunek odwiedzenia każdego miasta przynajmniej raz.

Problemy podzielić można na *decyzyjne*, *optymalizacyjne* oraz problem *przeszukiwania*.

Def. 2.3.2. Problem decyzyjny, [4]
Problemem decyzyjnym Π określamy taki problem, dla którego zdefiniowano pytanie na które należy dać odpowiedź TAK lub NIE.

Przykładem może tutaj być problem wspólnego superciągu, w którym instancją jest zbiór słów, a odpowiedź brzmi TAK wtedy i tylko wtedy, jeśli dla wszystkich słów instancji istnieje superciąg będący ich złożeniem, równy lub mniejszy pewnej wartości n określającej jego długość, danej w instancji problemu.

Def. 2.3.3. Problem optymalizacyjny, [4]
Problemem optymalizacyjnym Π określamy taki problem, dla którego wymagana jest ekstremalizacja pewnej funkcji celu.

Def. 2.3.4. Problem przeszukiwania
Problemem przeszukiwania Π określamy taki problem, dla którego należy odszukać kompletne rozwiązanie dla zadanych parametrów lub podać odpowiedź NIE jeżeli ono nie istnieje.

Przykładem może być odszukanie wspólnego superciągu mniejszego lub równego wartości parametru określonego w instancji problemu.

Klasy problemów decyzyjnych oraz optymalizacyjnych są ze sobą powiązanie w kontekście złożoności obliczeniowej. Jeśli z jakimś problemem optymalizacyjnym, tj. wymagającym ekstremalizacji pewnej funkcji celu związżemy odpowiadający mu problem decyzyjny, w którym pytamy o istnienie rozwiązania spełniającego dane warunki podane w instancji tego problemu, wtedy taki problem decyzyjny nie jest obliczeniowo trudniejszy niż jego odpowiednik optymalizacyjny [4], [38]. Uzasadnienie jest dość oczywiste: jeżeli w krótkim czasie jesteśmy w stanie rozwiązać problem optymalizacyjny, odpowiedź na pytanie problemu decyzyjnego jest w tym momencie równie prosta. Z drugiej strony, jeżeli problem decyzyjny jest "trudny"do rozwiązania (ponieważ np. udowodniono jego NP-zupełność), to odpowiadający mu problem optymalizacyjny jest co najmniej równie trudny. Implikacja tych zależności determinuje możliwe podejście do wykazywania stopnia złożoności problemów: aby wykazać łatwość problemu decyzyjnego wystarczy wykazać łatwość rozwiązania jego wersji optymalizacyjnej, jeśli zaś chcemy wykazać trudność problemu optymalizacyjnego, wystarczy wykazać trudność jego wersji decyzyjnej.

Z tematyką problemów obliczeniowych związane są też zasady zapisywania ich instancji, czyli tak zwane *reguły kodowania*. Dane konkretnej instancji I problemu Π należącej do zbioru wszystkich instancji omawianego problemu D_Π co oznaczamy $I \in D_\Pi$ koduje się za pomocą skończonego łańcucha $x(I)$ symboli należących do z góry określonego alfabetu Σ zgodnie z ustaloną reguła kodowania. Rozmiar konkretnego problemu $N(I)$ to długość łańcucha danych $x(I)$ [4]. Przyjąć dalej należy tak zwaną *rozsądną regułę kodowania* spełniającą dwa podstawowe warunki. Po pierwsze w łańcuchu kodującym dane nie może być symboli nadmiarowych, po drugie liczby powinny być zapisywane przy podstawie zliczania większej niż 1, np. binarnie.

Idąc dalej, *algorytmem* nazywać będziemy listę rozkazów (elementarnych kroków) do wykonania, która w rezultacie pozwala otrzymać rozwiązanie pewnego problemu. Algorytm A *rozwiązuje* problem Π, jeżeli można go zastosować do każdej instancji I problemu Π ($I \in D_\Pi$) w celu otrzymania rozwiązania tej instancji. Algorytm dla problemu charakteryzuje się generalnie trzema głównymi kryteriami, których spełnienie decyduje o jego poprawności - są to jednoznaczność, skończoność oraz kompletność. Jednoznaczność oznacza, że algorytm dla tych samych danych będzie generować te same wyniki. Skończoność wymaga aby algorytm był w stanie stworzyć rozwiązanie problemu w pewnej skończonej liczbie kroków. Kompletność algorytmu oznacza, że jest on w stanie rozwiązać każdą instancję I problemu Π taką, że $I \in D_\Pi$.

Mając dany pewien algorytm musimy dysponować narzędziem do jego oceny z punktu widzenia teorii złożoności obliczeniowej. Kolejne definicje umożliwią skonstruowanie takiego narzędzia.

Def. 2.3.5. Rząd funkcji, [4]
Funkcja $f(k)$ jest rzędu $g(k)$, co zapisujemy $O(g(k))$, jeżeli istnieje taka stała c, że

$|f(k)| \leqslant c|g(k)|$ dla prawie wszystkich wartości k

Def. 2.3.6. Funkcja złożoności obliczeniowej, [4]
Funkcja złożoności obliczeniowej algorytmu A rozwiązującego problem Π przyporządkowuje każdej wartości rozmiaru konkretnego problemu $I \in D_\Pi$ maksymalną liczbę kroków potrzebnych do rozwiązania za pomocą algorytmu A instancji problemu Π o tym rozmiarze, to znaczy:
$f_A(n) = max\{t : t$ jest liczbą elementarnych kroków maszyny cyfrowej niezbędną do rozwiązania problemu $I \in D_\Pi \wedge n = N(I)\}$.

Mając zdefiniowaną funkcję złożoności obliczeniowej można podzielić algorytmy na odpowiednie klasy. Popularny podział, którego implikacje będą wykorzystywane w niniejszej pracy to podział na *algorytmy wielomianowe* oraz *algorytmy wykładnicze*.

Def. 2.3.7. Podstawowy podział algorytmów, [4]
Algorytmem wielomianowym nazywamy algorytm, którego funkcja złożoności obliczeniowej jest $O(p(k))$, przy czym p jest pewnym wielomianem, a k rozmiarem konkretnego rozwiązywanego problemu. Każdy inny algorytm, którego funkcja złożoności obliczeniowej nie może być w ten sposób ograniczona nazywać będziemy *algorytmem wykładniczym*.

W dalszych rozważaniach wprowadzony zostanie pewien abstrakcyjny model maszyny cyfrowej. Celem jest uniezależnienie się od jakiegokolwiek typu komputera. Poniżej jako model obliczeń przedstawione zostaną dwie maszyny Turinga: deterministyczna oraz niedeterministyczna.

Deterministyczna maszyna Turinga (DTM) może być opisana jako 'urządzenie' składające się z nieskończenie długiej taśmy, na której zapisywane lub odczytywane są symbole, z głowicy odpowiedzialnej za zapis i odczyt mogącej poruszać się po taśmie oraz systemu sterującego głowicą, w zależności od określonego w danej chwili stanu maszyny. Maszynę Turinga dla pewnego programu określa się poprzez podanie:

1) skończonego zbioru symboli Γ zawierającego podzbiór $\Sigma \subset \Gamma$ tak zwanych *symboli wejściowych* oraz symbol pusty $b \in \Gamma - \Sigma$,

2) skończonego zbioru stanów Q, zawierającego wyróżniony stan początkowy q_0 oraz dwa stany końcowe q_Y oraz q_N,

3) funkcję przejść $\delta : (Q - \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$.

Zaczynając w stanie początkowym q_0 deterministyczna maszyna Turinga porusza głowicą zgodnie z regułami zdefiniowanymi przez funkcję przejść w zależności od tego, w

jakim jest w danym momencie stanie oraz na jakim symbolu znajduje się głowica. Wykonywanie programu trwa do chwili, w której DTM osiągnie jeden ze stanów końcowych q_Y lub q_N, przy czym odpowiednio odpowiedź brzmi wtedy TAK lub NIE.

Deterministyczna maszyna Turinga rozwiązuje problem decyzyjny Π przy kodowaniu e, jeżeli zatrzymuje się dla wszystkich łańcuchów wejściowych i kończy obliczenia w stanie q_Y dla wszystkich instancji rozpatrywanego problemu należących do zbioru instancji dla których odpowiedź brzmi TAK i tylko dla nich [4]. Jak widać jest to działanie mające na celu rozwiązanie problemu decyzyjnego, który z definicji takiej właśnie odpowiedzi wymaga jako swojego rozwiązania.

Kończąc wywód dotyczący maszyny deterministycznej, należy także wspomnieć o istnieniu modelu tzw. *k-taśmowej deterministycznej maszyny Turinga*, która jak łatwo się domyślić posiada k taśm (oraz k głowic pracujących na odpowiednich taśmach). Model ten w niniejszej rozprawie nie będzie jednak dalej wykorzystywany.

Kolejnym przedstawionym modelem obliczeń będzie *niedeterministyczna maszyna Turinga* (NDTM), którą można sobie wyobrazić jako DTM wyposażoną w moduł generujący. W pierwszym etapie moduł ten generuje całkowicie losowo łańcuch S symboli ze zbioru Γ. W drugim etapie maszyna zachowuje się jak DTM, sprawdzając czy wygenerowany łańcuch spełnia warunki określone w pytaniu dla problemu. Niedeterministyczna maszyna Turinga rozwiązuje pewien problem decyzyjny Π, jeżeli dla każdego $I \in D_\Pi$ spełnione są warunki:

1) jeśli odpowiedź brzmi TAK, to zostanie wygenerowany łańcuch S, który spowoduje, że po wykonaniu programu NDTM zakończy pracę w stanie q_Y,

2) jeśli odpowiedź brzmi NIE wtedy NDTM dla każdego wygenerowane łańcucha osiąga stan q_N lub nie zakończy pracy w skończonym czasie.

Podobnie jak poprzednio istnieje także model *k-taśmowej NDTM*, nie będzie on jednak dalej rozważany. Poniższe twierdzenie pokazuje związek między DTM a NDTM w kontekście złożoności problemów:

Twierdzenie 2.3.1. [4]
Jeżeli jednotaśmowa niedeterministyczna maszyna Turinga rozwiązuje problem decyzyjny π w czasie wielomianowym, to istnieje wielomian p taki, że jednotaśmowa deterministyczna maszyna Turinga rozwiązuje ten problem w czasie $O(2^{p(N(I))})$, gdzie $I \in D_\pi$.

Klasy złożoności obliczeniowej umożliwiają klasyfikację problemów decyzyjnych. Klasę **P** tworzą wszystkie problemy decyzyjne, które deterministyczna maszyna Turinga rozwiązuje w co najwyżej wielomianowym czasie. Klasę **NP** tworzą natomiast wszystkie

problemy decyzyjne które w co najwyżej wielomianowym czasie rozwiązuje niedeterministyczna maszyna Turinga.

Problem zależności między tymi dwoma klasami pozostaje do dziś nierozstrzygnięty, z przyczyn mających jak najbardziej praktyczne zastosowanie przyjmuje się jednak, że $\mathbf{P} \neq \mathbf{NP}$. Z powyższego założenia wynika więc, że problemy należące do **NP**, lecz nie należące do klasy **P** charakteryzują się tym, że o ile ich rozwiązania są weryfikowalne w czasie wielomianowym, to nie wiadomo czy istnieją dla takich problemów wielomianowe algorytmy, które byłyby w stanie te problemy rozwiązać. Jeśli o problemie wiadomo więc tylko tyle, że należy do **NP**, to oznacza to, że problem ten może rozwiązać w czasie wykładniczym niedeterministyczna maszyna Turinga. Deterministyczna maszyna Turinga może co najwyżej w wielomianowym czasie zweryfikować rozwiązanie.

Jeżeli przyjmujemy własność $\mathbf{P} \subseteq \mathbf{NP}$, oznacza to, że do klasy **NP** należą problemy zarówno "łatwe"(rozwiązywane przez algorytmy wielomianowe) jak i "trudne"(dla których algorytmy wielomianowe mogą nie istnieć). Do dalszych rozważań potrzebne jest zdefiniowanie tzw. *transformacji wielomianowej.*

Def. 2.3.8. Transformacja wielomianowa, [4]
Transformacją wielomianową problemu Π_1 do problemu Π_2 ($\Pi_1 \propto \Pi_2$) nazywamy funkcję $f : D_{\Pi_1} \rightarrow D_{\Pi_2}$ spełniającą warunki:

1) dla każdego $I_{\in} D_{\Pi_1}$ odpowiedź brzmi TAK wtedy i tylko wtedy, gdy dla $f(I)$ odpowiedź brzmi również TAK.

2) czas obliczania funkcji f przez DTM dla każdego $I \in D_{\Pi_1}$jest ograniczony od góry przez wielomian od $N(I)$.

Problem decyzyjny Π_2 jest NP–zupełny, jeżeli $\Pi_2 \in \mathbf{NP}$ i dla każdego problemu decyzyjnego $\Pi_1 \in \mathbf{NP}$, $\Pi_1 \propto \Pi_2$ [4]. Z powyższej definicji nasuwa się wniosek, że gdyby istniał algorytm wielomianowy rozwiązujący jakikolwiek problem NP-zupełny, to każdy problem z klasy **NP** (a więc także wszystkie problemy NP-zupełne) można by rozwiązać wielomianowo. Z definicji transformacji wielomianowej wynika też drugi wniosek, tym razem bezpośrednio związany z udowadnianiem NP-zupełności problemów decyzyjnych. Aby wykazać NP-zupełność jakiegoś problemu decyzyjnego, należy transformować do niego wielomianowo jakiś inny, znany problem NP-zupełny. Podejście to zakłada znajomość pewnego podstawowego problemu NP-zupełnego, transformującego się do innych znanych problemów z tej klasy. Taki problem to problem *spełnialności wyrażeń boolowskich (SAT)*, którego dowód złożoności, tj. precyzyjniej: jego przynależność do klasy problemów NP–zupełnych przedstawił Stephen Cook w pracy [12].

Ostatnią klasą złożoności ważną dla niniejszej rozprawy, która zostanie teraz omówiona jest klasa problemów *silnie NP-zupełnych.* Wymaga ona wprowadzenia definicji

algorytmu pseudowielomianowego.

Def. 2.3.9. Algorytm pseudowielomianowy, [4]
Algorytmem pseudowielomianowym nazywamy taki algorytm, którego funkcja złożoności obliczeniowej jest ograniczona od góry przez wielomian zależny od rozmiaru konkretnego problemu $N(I)$ oraz największej liczby $Max(I)$ występującej w jego instancji.

Korzyść ze stosowania tego typu algorytmów wynika z faktu, że wartość największej liczby w instancji nie jest w niektórych praktycznych problemach na tyle duża, aby znacząco wpłynąć na realistyczny czas obliczeń. Algorytm pseudowielomianowy można skonstruować tylko dla problemów decyzyjnych liczbowych Π, to znaczy takich, dla których nie istnieje wielomian p taki, że $Max(I) \leqslant p(N(I))$ dla każdego $I \in D_\pi$. Można teraz przytoczyć dwie definicje, pozwalają doprecyzować ideę silnej NP-zupełności.

Def. 2.3.10. [4]
Dla dowolnego problemu decyzyjnego Π i dowolnego wielomianu p określonego dla liczb całkowitych, niech Π_p oznacza podproblem Π otrzymany przez ograniczenie D_Π tylko do tych instancji, dla których $Max(I) \leqslant p(N(I))$. Zatem taki podproblem Π_p nie jest problemem liczbowym.

Def. 2.3.11. Silna NP-zupełność problemów decyzyjnych, [4]
Problem decyzyjny Π jest *silnie NP-zupełny*, jeśli należy do **NP** i istnieje wielomian p określony dla liczb całkowitych, dla którego Π_p jest NP-zupełny.

Taka definicja silnej NP-zupełności wymaga znalezienia wielomianu p o którym wciąż mowa. Dlatego zamiast tej drogi, można w udowadnianiu silnej NP-zupełności posługiwać się transformacją pseudowielomianową:

Def. 2.3.12. Transformacja pseudowielomianowa, [4]
Pseudowielomianową transformacją problemu Π_1 do problemu Π_2 nazywamy funkcję $f : D_{\Pi_1} \to D_{\Pi_2}$, taką, że:

1) dla każdego konkretnego problemu $I \in D_{\Pi_1}$ odpowiedź brzmi TAK wtedy i tylko wtedy, gdy dla $f(I)$ odpowiedź także brzmi TAK;

2) funkcja f może być obliczona przez DTM w czasie ograniczonym od góry przez wielomian zależny od dwóch zmiennych: $Max(I)$ i $N(I)$;

3) istnieje wielomian q_1, taki, że dla każdego $I \in D_{\Pi_1}$:
$q_1(N(f(I))) \geqslant N(I))$

4) istnieje wielomian dwóch zmiennych q_2, taki, że dla każdego $I \in D_{\Pi_1}$ zachodzi: $Max(f(I)) \leqslant q_2(Max(I), N(I))$.

W zamian za osłabienie warunku wielomianowego czasu realizacji (por. transformacja wielomianowa) wymagane jest aby rozmiar konkretnego problemu nie zmalał wykładniczo, oraz aby wykładniczo nie wzrosła największa wartość liczbowa zawarta w danych [4].

Def. 2.3.13. Silna NP-zupełność problemu, [20], [4]
Problem Π jest silnie NP-zupełny jeżeli należy do klasy **NP** i istnieje inny problem silnie NP-zupełny, który pseudowielomianowo transformuje się do problemu Π.

Na zakończenie należy wprowadzić zagadnienie NP-trudności oraz silnej NP-trudności.

Def. 2.3.14. Silna NP-zupełność problemu, [20], [4]
Wielomianową transformacją Turinga problemu przeszukiwania Π_1 do problemu przeszukiwania Π_2 nazywamy algorytm A rozwiązujący problem Π_1 oraz używający pewnej hipotetycznej procedury S rozwiązującej Π_2 w ten sposób, że jeżeli S byłby algorytmem wielomianowym dla rozwiązania Π_2, wtedy A byłby wielomianowym algorytmem do rozwiązywania problemu Π_1.

Podobnie więc jak przy poprzedniej definicji silnej NP-zupełności problemu, możemy sformułować definicję następującą.

Def. 2.3.15. NP-trudność problemu, [20], [4]
Problem przeszukiwania Π_2 jest NP-trudny jeżeli istnieje NP-zupełny problem Π_1 dla którego można użyć wielomianowej transformacji Turinga w celu otrzymania problemu Π_2.

Z definicji wynika więc, że każdy problem NP-zupełny jest równocześnie NP-trudny [20]. Jeżeli chodzi o silną NP-trudność, to jej dowodzenie jest podobne z tą oczywistą różnicą, że transformacja musi użyć problemu silnie NP-zupełnego.

Rozdział 3

Podstawowe zagadnienia biologii molekularnej

3.1 Budowa DNA

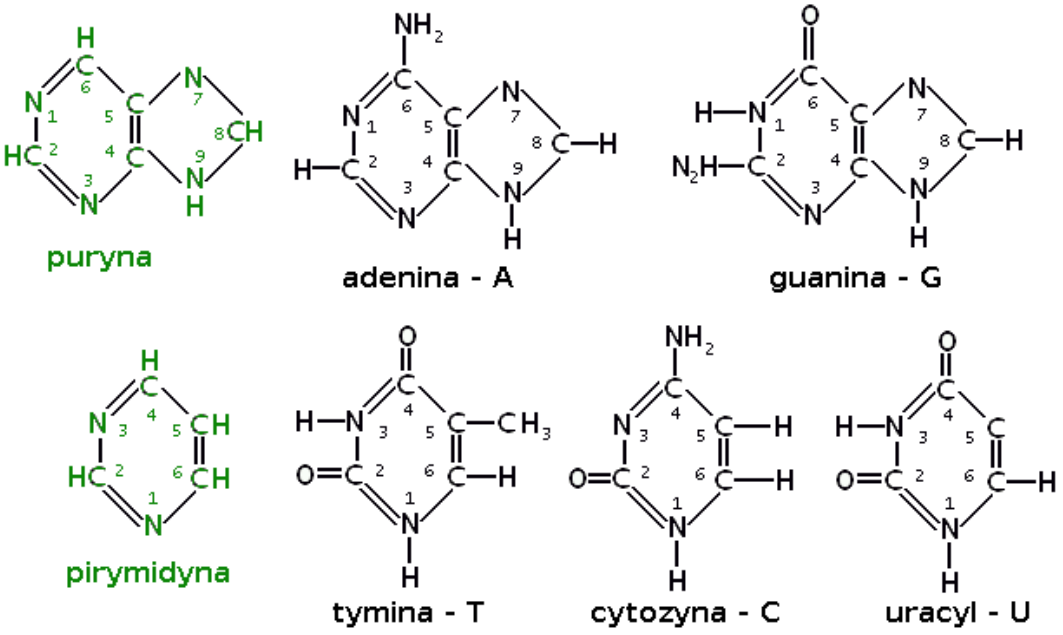
W niniejszym rozdziale przedstawione zostaną zagadnienia dotyczące budowy nukleotydów, istotne z punktu widzenia całej pracy. Najważniejszym tematem, który zostanie w tej sekcji precyzyjnie przedstawiony jest budowa kwasu deoksyrybonukleinowego (DNA). Jak napisano na wstępie, struktura DNA została po raz pierwszy opisana w czasopiśmie *Nature* w roku 1953 [53]. Autorami tej pracy byli James D. Watson oraz Francis Crick. Wiedza płynąca z opublikowanego w artykule odkrycia okazała się prawdziwie przełomowa, gdyż umożliwiła dalszy rozwój nauk mających na celu poznanie budowy DNA.

Obok DNA przechowującego informację genetyczną w żywych organizmach występuje także cząsteczka bardzo doń podobna - kwas rybonukleinowy (RNA). Pomimo pewnych podobieństw jest on osobną jednostką w komórkach. Różnice są widoczne między innymi w samej budowie obu kwasów. Podczas gdy DNA występuje jako podwójna helisa, RNA jest zazwyczaj jednoniciowe. Jeśli w uproszczeniu określimy DNA jako uporządkowaną sekwencje genów niczym encyklopedię z kolejnymi hasłami, wtedy RNA można nazwać roboczymi notatkami niezbędnymi do wykorzystania wiedzy z przytoczonej tu w przykładzie encyklopedii (DNA). RNA jest między innymi odpowiedzialne za przekazywanie informacji z DNA niezbędnych w procesie tworzenia nowych białek, funkcje regulujące prace komórki jak np. ekspresja genów. W tematyce RNA w ostatnich latach następuje wiele znaczących odkryć. RNA pełni w komórkach o wiele więcej funkcji niż pierwotnie zakładano, a badania nad różnymi jego podtypami jak na przykład microRNA czy siRNA są współcześnie intensywnie rozwijane.

Zarówno DNA jak i RNA są polimerami, które zbudowane są z mniejszych jednostek

zwanych nukleotydami. W nukleotydach można wyróżnić trzy części – związki chemiczne o konkretnych właściwościach: zasadę azotową, cukier zwany rybozą lub deoksyrobozą zależnie czy nukleotyd jest częścią RNA czy DNA oraz grupę fosforanową.

Zasady azotowe spotykane w nukleotydach są pochodnymi puryny lub pirymidyny – związków heterocyklicznych. Same związki heterocykliczne to takie, których pierścienie zbudowane są oprócz węgla także z atomów innych pierwiastków. Tak dokładny opis budowy zawarty w niniejszym rozdziale okaże się przydatny w dalszej części pracy, kiedy poszczególne nukleotydy będą grupowane do pewnych zbiorów ze względu na swoje właściwości w procesach sekwencjonowania DNA. Ich właściwości biorą się właśnie z przedstawianej tutaj budowy molekularnej. Podział ze względu na zasady azotowe budujące część struktury nukleotydu w kwasie deoksyrobonukleinowym przedstawia Rysunek 3.1:

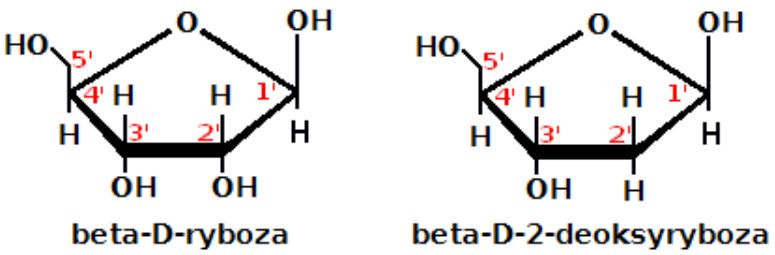


Rysunek 3.1: Zasady azotowe nukleotydów

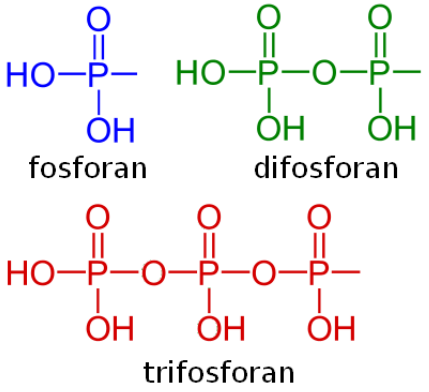
Kolejnym składnikiem nukleotydów są cukry. W kwasach nukleinowych spotyka się ich dwa rodzaje: beta-d-rybozę oraz beta-d-2-deoksyrybozę. W zależności od rodzaju cukru mamy doczynienia z RNA (ryboza) lub DNA (deoksyryboza). Cukry te pokazano na Rysunku 3.2.

Trzecim składnikiem budującym nukleotydy są fosforany. Występują one w wielu związkach chemicznych, między innymi jako monofosforany, difosforany lub trifosforany. Nukleotydy budujące DNA i RNA zawierają w sobie monofosforany należące do grupy fosforanowej, której elementy zostały pokazane Rysunku 3.3.

Omówione tutaj elementy składowe budują 4 podstawowe nukleotydy w DNA których typ zależy od zasady azotowej: nukleotyd A (adenina), nukleotyd C (cytozyna),

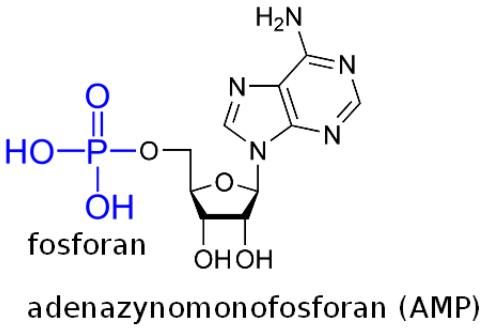


Rysunek 3.2: Cukry



Rysunek 3.3: Fosforany

nukleotyd G (guanina) oraz nukleotyd T (tymina). Naczelnym tematem pracy jest badanie sekwencji DNA, jednak wypada w tym miejscu nadmienić, że różnica w budowie molekularnej między DNA a RNA jest również taka, że w tym drugim zamiast tyniny znajduje się inna zasada azotowa - uracyl (nukleotyd U, zbudowany na bazie pirymidyny). Na Rysunku 3.4 przedstawiono przykładowo pełny schemat budowy nukleotydu typu A czyli połączenia zasady azotowej z cukrem (tworzących tzw. nukleozyd) oraz fosforanu.

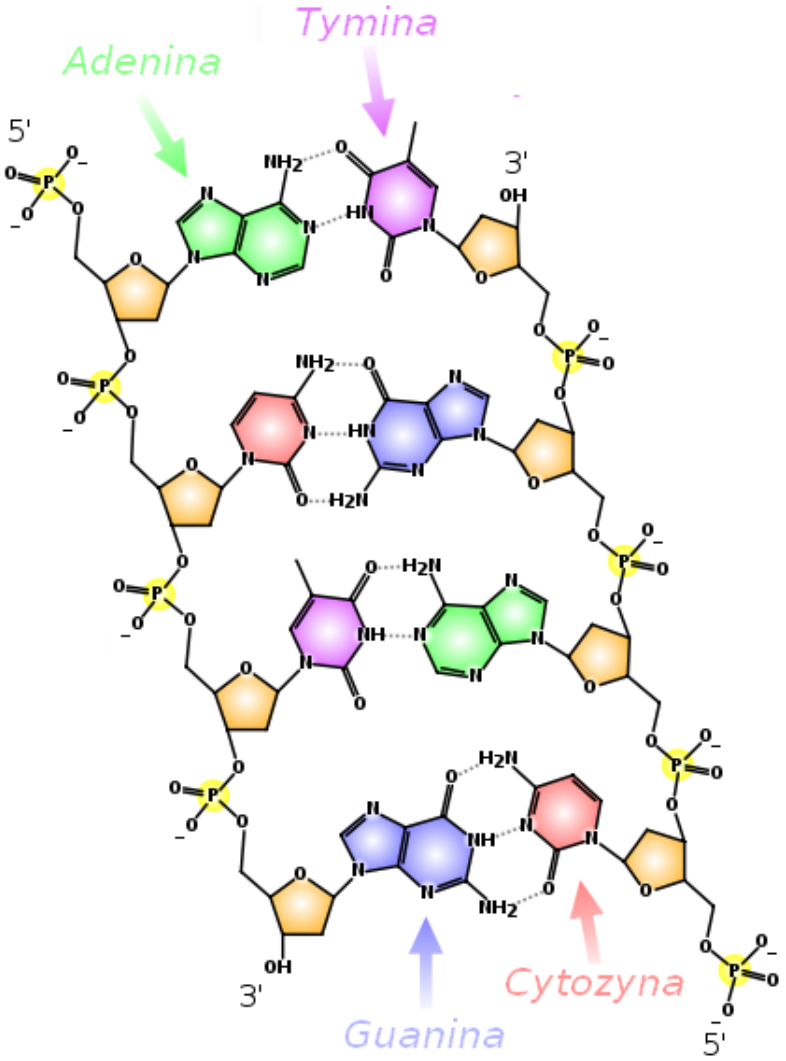


Rysunek 3.4: Nukleotyd A

Kolejnym zagadnieniem, mającym istotne znaczenie, między innymi dla sekwencjonowania przez hybrydyzację, jest *komplementarność* nukleotydów. Jak napisane zostało we wstępie, zbudowane z dwóch łańcuchów DNA ma kształt tak zwanej podwójnej helisy. Jej konstrukcja z punktu widzenia połączeń między dwoma łańcuchami jest niezmiernie prosta: jeśli w pewnym miejscu znajduje się nukleotyd A, na drugim łańcuchu znajduje się nukleotyd T. Jeśli mówimy o nukleotydzie C wtedy komplementarnym do niego jest nukleotyd G. Komplementarność zależy od rodzaju wiązania między dwoma nukleotydami znajdującymi się na dwóch niciach. Wiązania te są na tyle silne, że w przypadku rozdzielenia łańcuchów będą one mogły złączyć się ponownie, jeśli tylko zapewnione zostaną odpowiednie warunki temperatury i zasolenia roztworu z DNA. Precyzyjniej, wiązanie o którym tutaj mowa nosi nazwę wiązania wodorowego. Adenina z tyminą połączone są podwójnym mostkiem wodorowym, cytozyna z guaniną - potrójnym. To rozróżnienie jest powodem przyjętego w dziedzinie badań nad DNA podziału wiązań na słabe (A-T) oraz silne (C-G), co również ma istotne znaczenie w metodach sekwencjonowania.
Drugim ważnym rodzajem wiązań są tzw. wiązania fosfodiestrowe. Takie wiązania łączą kolejne nukleotydy w ramach jednej nici. Numerując atomy węgla w opisanych związkach cyklicznych w wiązaniach tych biorą udział trzeci atom jednego nukleotydu oznaczany przez 3' (precyzyjnie: atom związku heterocyklicznego nukleotydu) oraz piąty drugiego - 5' (biorąc pod uwagę pewną połączoną parę nukleotydów). Na jednym końcu pełnej nici DNA znajduje się więc niewykorzystane wiązanie 3', a na drugim 5', co jest powodem istnienia tzw. polarności nici DNA. Należy tu także nadmienić, że w podwójnej nici tworzącej jedną cząstkę DNA nici te są skierowane w przeciwne strony z punktu widzenia ustawienia wiązań 3' oraz 5'. Na Rysunku 3.5 przedstawiono strukturę przykładowej cząsteczki DNA.

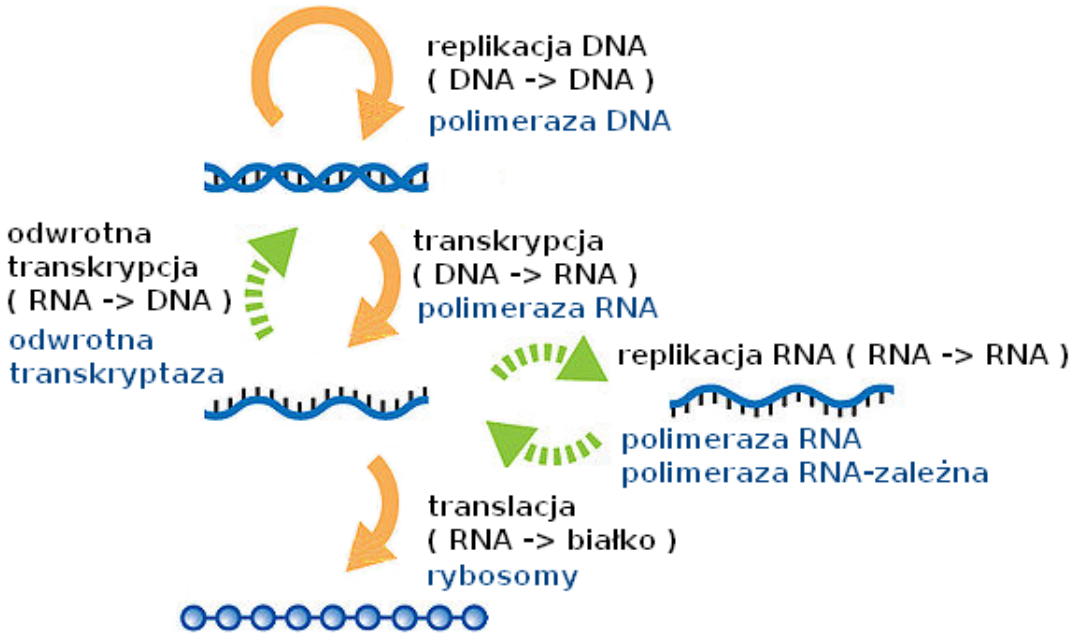
Kolejnym i ostatnim omawianym tutaj zagadnieniem jest tak zwany *centralny dogmat biologii molekularnej* opublikowany w [13, 14]. Jego implikacje nie dotyczą bezpośrednio metody sekwencjonowania będącej przedmiotem niniejszej rozprawy, stanowią jednak uzasadnienie ważności zagadnienia sekwencjonowania DNA jako takiego. Treścią centralnego dogmatu jest transfer informacji genetycznej. Dalej transfer ten pomiędzy DNA, RNA a białkiem nazywać będziemy przejściem. Na Rysunku 3.6 oznaczony został przez żółte i zielone strzałki.

Informacje zawarte w DNA są przepisywane w jednoniciowe RNA (w procesie transkrypcji przy udziale enzymu tzw. polimerazy RNA). RNA stanowi następnie bazę do budowy białek w procesie translacji z udziałem rybosomów. Poza tymi dwoma przejściami istnieje jeszcze proces tak zwanej replikacji DNA, w którym przy udziale enzymu polimerazy DNA, cząsteczka DNA może się powielić, co ma fundamentalne znaczenie dla rozmnażania komórek i organizmów.



Rysunek 3.5: Przykładowa cząsteczka DNA, [31]

Te trzy przejścia / transfery informacji genetycznej występują w komórkach w naturalnych procesach życiowych. Istnieją kolejne trzy, które mogą wystąpić w specjalnych okolicznościach i są to: odwrotna transkrypcja (RNA w DNA), replikacja RNA (RNA w RNA) oraz bezpośrednia translacja informacji zawartej w DNA w białko z pominięciem fazy transkrypcji DNA w RNA. Pierwsze dwa sposoby transferu informacji występują w wirusach, trzeci okazał się możliwy w badaniach laboratoryjnych z pominięciem komórek organizmów [36], [52]. Jak dotąd nigdzie w naturze nie zaobserwowano przejścia białek w DNA czy RNA, co według centralnego dogmatu uznawane jest za niemożliwe. Co do przejścia informacji z białek w białka, można ostrożnie powiedzieć, że jest to raczej mało prawdopodobne, aczkolwiek problem jest nierozstrzygnięty. Odpowiadają za to białka infekcyjne zwane prionami, co do których istnieją hipotezy mówiące, że mogą one być pierwszym znanym czynnikiem infekcyjnym, który działa bez udziału kwasów nukleinowych.



Rysunek 3.6: Przejścia w Centralnym Dogmacie biologii molekularnej, [25]

Rozdział 4

Sposoby sekwencjonowania DNA, sekwencjonowanie przez hybrydyzację

4.1 Wybrane metody sekwencjonowania DNA

4.1.1 Metoda Sangera, sekwencjonowanie dideoxowe

Metoda ta opracowana została przez Fredericka Sangera, za którą w roku 1980 otrzymał on nagrodę Nobla. Jej liczne modyfikacje sprawiają, że do dnia dzisiejszego jest używana w procesie sekwencjonowania [46], [47].

Metoda ta wykorzystuje nukleotydy dideoksyrybonukleinowe, zwane dalej dideoksynukleotydami (ddNTP). Możliwe jest uzyskanie dideoksynukleotydu dla każdego nukleotydu standardowego A, C, G, T występującego w kwasie DNA. Zasadnicza różnica pomiędzy ddNTP a normalnym nukleotydem polega na posiadaniu przez te pierwsze grupy hydrogenowej na końcu 3' zamiast standardowej grupy -OH. Ta modyfikacja powoduje, że niemożliwe staje się dołączenie kolejnego nukleotydu do nukleotydu typu ddNTP, co kończy reakcję syntezy DNA przy ich użyciu.

Przed rozpoczęciem sekwencjonowania DNA opisywaną metodą należy przygotować wzorzec DNA, czyli jednoniciowe DNA, którego kolejność nukleotydów chcemy poznać. Do wzorca dołączany jest starter DNA w miejscu, od którego chcemy zacząć sekwencjonowanie. Proces syntezy, czyli tworzenia DNA przy użyciu wzorca jest dokonywany niezależnie od siebie w 4 wersjach, różniących się typem dideoksynukleotydu zawartego w roztworze. Skład chemiczny czterech 'probówek' używanych do syntezy można przedstawić następująco:

1. typ "A": 4 normalne nukleotydy, ddATP, polimeraza DNA;
2. typ "C": 4 normalne nukleotydy, ddCTP, polimeraza DNA;
3. typ "G": 4 normalne nukleotydy, ddGTP, polimeraza DNA;
4. typ "T": 4 normalne nukleotydy, ddTTP, polimeraza DNA.

Przyjmuje się, że stosunek koncentracji nukleotydów ddNTP do ich 'naturalnych' odpowiedników w roztworze to 1:100. Dzięki temu przy stosowaniu metody nie powstaje mnóstwo najkrótszych możliwych fragmentów DNA, zsyntetyzowanych według danej matrycy, kończących się pierwszym wystąpieniem odpowiedniego typu nukleotydu, którego 'terminator' jest w roztworze. W efekcie powstaje za to duża liczba fragmentów DNA o różnej długości, zależnie od tego, kiedy nukleotyd ddNTP został doczepiony do łańcucha, kończąc dalszą syntezę. Z uwagi na wysokie prawdopodobieństwo błędów, w metodzie odrzuca się najdłuższe otrzymane łańcuchy DNA.

Poniższy przykład pokazuje możliwe wyniki reakcji syntezy w pewnej probówce typu "G":

```
      5'-GAATGTCCTTTCTCTAAGTCCTAAG
3'-GGAGACTTACAGGAAAGAGATTCAGGATTCAGGAGGCCTACCATGAAGATCAAG-5'

      5'-GAATGTCCTTTCTCTAAGTCCTAAGTCCTCCG
3'-GGAGACTTACAGGAAAGAGATTCAGGATTCAGGAGGCCTACCATGAAGATCAAG-5'

      5'-GAATGTCCTTTCTCTAAGTCCTAAGTCCTCCGG
3'-GGAGACTTACAGGAAAGAGATTCAGGATTCAGGAGGCCTACCATGAAGATCAAG-5'

      5'-GAATGTCCTTTCTCTAAGTCCTAAGTCCTCCGGATG
3'-GGAGACTTACAGGAAAGAGATTCAGGATTCAGGAGGCCTACCATGAAGATCAAG-5'

      5'-GAATGTCCTTTCTCTAAGTCCTAAGTCCTCCGGATGG
3'-GGAGACTTACAGGAAAGAGATTCAGGATTCAGGAGGCCTACCATGAAGATCAAG-5'

      5'-GAATGTCCTTTCTCTAAGTCCTAAGTCCTCCGGATGGTACTTCTAG
3'-GGAGACTTACAGGAAAGAGATTCAGGATTCAGGAGGCCTACCATGAAGATCAAG-5'
```

Rysunek 4.1: Przykład syntezy w metodzie Sangera

W momencie, kiedy reakcje ze wszystkich czterech próbek zostają zakończone, ich produkty są przygotowywane do elekroforezy żelowej. Zawartość każdej z probówek jest umieszczana na żelu, aby oddzielić od siebie fragmenty o różnych długościach. Użycie wysokorozdzielczej elektroforezy żelowej umożliwia oddzielenie od siebie fragmentów różniących się długością z dokładnością do jednego nukleotydu. Następnie żel jest naświetlany promieniami UV lub Röntgena w zależności od metody znakowania DNA z pierwszego kroku metody, kiedy to przygotowywana była matryca sekwencji. W tej fazie możliwe jest już uzyskanie obrazu pokazującego oznakowane wcześniej nukleotydy

ddNTP - terminatory reakcji syntezy DNA, w różnych odległościach od początku sekwencji (tj. startera). Dzięki tego uzyskiwany jest autoradiogram, czyli obraz występowania kolejności nukleotydów w badanym DNA. Jego odczytanie jest końcowym krokiem metody, dzięki czemu poznawana jest kolejność występowania nukleotydów w badanej sekwencji DNA.

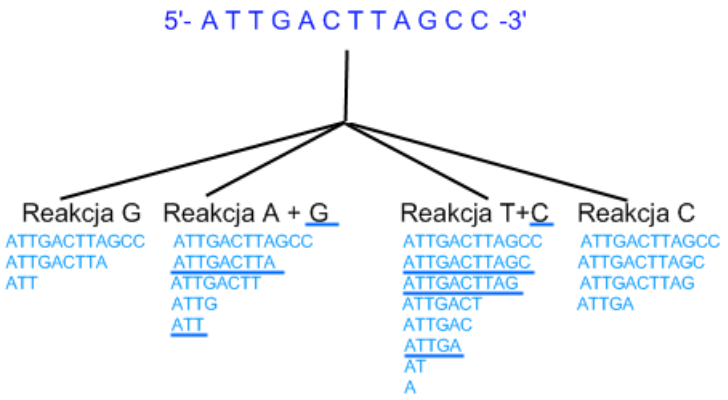
4.1.2 Metoda Maxama-Gilberta

Metoda Maxama-Gilberta została odkryta w latach 1976-1977 przez Allana Maxama i Waltera Gilberta [35], zwana jest także w literaturze jako sekwencjonowanie chemiczne. Współcześnie jest to metoda rzadko stosowana głównie z powodu skali trudności jej przeprowadzenia, co pociąga za sobą koszty powstałe w wyniku używania mało bezpiecznych związków chemicznych. Metodę tę trudno używać do sekwencjonowania dłuższych fragmentów DNA.

Metoda ta wykorzystuje specjalnie dobrane związki chemiczne do przeprowadzania cięć odpowiedniego typu (G, A+G, T+C, C), dzięki czemu otrzymywane są fragmenty badanego DNA kończące się odpowiednim, znanym typem nukleotydu. W przypadku reakcji A+G oraz T+C, oprócz oczekiwanych produktów cięcia A i T powstają także odcięte fragmenty poprzez reakcje G (w A+G) i C (w T+C).

W ramach przygotowań do przeprowadzenia sekwencjonowania opisywaną metodą należy przygotować kopie łańcuchów DNA oznaczając je markerem na końcu 5', zazwyczaj za pomocą radioaktywnego fosforu. Następnie tak przygotowane fragmenty badanego DNA są wystawiane na działanie związków chemicznych powodujących dany typ cięcia z czterech wcześniej wymienionych. W efekcie następują odcięcia w danych fragmentach DNA, gdzie występował odpowiedni dla danej reakcji typ nukleotydu. Pod koniec eksperymentu dla wszystkich czterech reakcji otrzymujemy zbiór pociętych fragmentów DNA o różnej długości, które można uszeregować jak w poprzednio opisanej metodzie, za pomocą elektroforezy żelowej. Dzięki temu możliwe jest ich 'ustawienie' w zależności od długości. Poprzez naświetlanie żelu za pomocą promieniowania rentgenowskiego uzyskiwany jest autoradiograf. Paski na nim pozwalają odczytać kolejność występowania odpowiednich typów nukleotydów względem siebie, co efektywnie prowadzi do poznania kolejności, czyli sekwencji nukleotydów w DNA. Rysunek 4.2 przedstawia produkty odpowiednich reakcji dla przykładowego fragmentu DNA. W reakcjach A+G oraz T+C podkreślone zostały produkty cieć, odpowiednio G oraz C. Porównując te zbiory z produktami "czystych" reakcji typu G i C (pierwsza i ostatnia na rysunku) możliwe jest ustalenie produktów cięć A oraz T.

Główna różnica pomiędzy metodą Sangera a metodą Maxama-Grahama polega na tym, że w przypadku tej pierwszy DNA jest syntetyzowane przy użyciu odpowiedniego



Rysunek 4.2: Przykład cięć w metodzie Maxama-Gilberta

wzorca DNA, natomiast w przypadku opisywanej metody DNA jest przycinane w danych fragmentach, gdzie występuje nukleotyd związany z odpowiednim typem reakcji tnącej. Obie metody mają zasadniczą wadę ujawniającą się przy odczycie pociętych sekwencji. Fragmenty DNA, w których występuje duża liczba powtórzeń tego samego nukleotydu są trudne do odczytania z uwagi na problem określenia precyzyjnej liczby takich powtarzających się nukleotydów, zanim pojawi się za nimi nukleotyd innego typu. Problem dużej liczby powtórzeń tego samego nukleotydu w sekwencji DNA nie dotyczy tylko opisanych tutaj dwóch metod. Pojawia się on nawet w najnowszych podejściach do sekwencjonowania.

4.1.3 Pirosekwencjonowanie, 454

Jest to metoda sekwencjonowania DNA bazująca na sekwencjonowaniu przez syntezę, tak jak opisana wcześniej metoda Sangera. W pirosekwencjonowaniu jednak nie porównuje się długości zbudowanych fragmentów przez elektroforezę żelową i autoradiogram, lecz wykrywany jest sam moment dołączenia kolejnego nukleotydu do sekwencji w czasie jej konstruowania według wzorca badanego DNA [34, 54]. W metodzie Sangera wykrywany jest dideoksynukleotyd kończący budowaną sekwencję. W metodzie pirosekwencjonowania wykrywane są pirofosforany wyzwalane w reakcji dołączania "nukleotydów" (precyzyjniej: trifosforanów deoksynukleotydów) do budowanej sekwencji.

Jednym z przykładów użycia tego podejścia jest jego zastosowanie w sekwenatorach DNA firmy 454 Life Science. W procesie wstępnym badane DNA jest cięte na części a następnie dołączane do tak zwanych koralikowych płaszczy streptawidynowych. Jednoniciowy wzorzec DNA (single-stranded template of DNA, sstDNA) jest unieruchamiany na wspomnianym koraliku, a następnie całość poddawana jest reakcji PCR do wytworzenia na nim kopii wzorca.

Koraliki te (każdy zawierający kopie innego fragmentu pociętego DNA) są następnie

umieszczane w tak zwanych studniach na specjalnym urządzeniu. W czasie trwania eksperymentu dostarczane są cyklicznie wszystkie nukleotydy potrzebne do syntezy DNA według danego w studni wzorca. Jednym z enzymów katalizujących opisywaną reakcję jest lucyferaza znana ze zjawiska bioluminescencji. Dzięki temu w momencie dołączenia odpowiedniego nukleotydu do syntetyzowanej według wzorca sekwencji na koraliku, specjalna kamera jest w stanie wykryć dane dołączenie, rejestrując je i zapisując, dzięki czemu możliwe staje się odczytanie sekwencji DNA. Ponieważ kolejne nukleotydy są dostarczane cyklicznie, w danym kroku reakcja syntezy może być jednoznacznie zidentyfikowana, tj. nukleotyd, który w danym koraliku ją spowodował.

4.1.4 Sekwenator firmy Illumina/Solexa

Metoda sekwencjonowania zastosowana w sekwenatorze firmy Illumina to w ogólności sekwencjonowanie przez syntezę. Na specjalnie przygotowanej płytce umieszczane są jednoniciowe fragmenty DNA, które następnie są zaginane w mostek. Mostek ten jest przyłączony do powierzchni za pomocą przygotowanych starterów dołączonych do fragmentów DNA. Następnie następuje powielanie fragmentów DNA w celu otrzymania takiej ich liczby w klastrach (grupach) dwuniciowych fragmentów DNA, która umożliwi otrzymanie sygnału o odpowiedniej sile w dalszej fazie sekwencjonowania. Siła sygnału jest istotną kwestią, kiedy następować będzie dobudowywanie sekwencji komplementarnych do tych dołączonych już do powierzchni. Dobudowanie to musi zostać poprawnie zarejestrowane przez odpowiednie kamery sekwenatora. Sam proces sekwencjonowania, czyli odczytywania fragmentów DNA w sekwenatorze odbywa się w kolejno następujących po sobie fazach. W każdej fazie na powierzchnię z fragmentami DNA doprowadzany jest roztwór wszystkich czterech nukleotydów, z czego każdy z nich ma dołączony odpowiedni znacznik, zależny od rodzaju nukleotydu. Dzięki zastosowaniu znaczników, w momencie, w którym nukleotyd danego typu dołącza się do przyczepionego na płytce DNA budując fragment komplementarny, zdarzenie takie jest wykrywane w czasie fotografowania płytki. Następnie wprowadzany jest związek chemiczny wymywający użyte znaczniki. Po tym następuje kolejna faza wprowadzania roztworu nukleotydów, w której dobudowywane one są do klastrów nici DNA. W momencie, w którym następuje przyłączanie odpowiednich nukleotydów w klastrach, tworzone są kolejne fotografie. Zawierają one obraz identyfikujący konkretne nukleotydy w danych klastrach. Zbiór fotografii rejestrujących syntezę z kolejnych faz, pozwala odtwarzać badane sekwencje DNA w sekwenatorze [32].

4.2 Sekwencjonowanie przez hybrydyzację - opis metody oraz wybrane podejścia do problemu

4.2.1 Podejście klasyczne

Klasyczne sekwencjonowanie przez hybrydyzację zostało przedstawione pod koniec lat 80tych, w pracach [1, 16, 27, 39]. Metoda ta wykorzystuje komplementarność nukleotydów budujących podwójną helisę DNA. Jak zostało już opisane w rozdziale dotyczącym zagadnień biochemicznych, cztery nukleotydy: A, C, G oraz T budujące DNA łączą dwie jego nici tylko poprzez pary A-T oraz C-G. Jeżeli na przykład w jednej nici znajduje się nukleotyd A, po drugiej stronie połączyć się on może tylko z nukleotydem T (oraz odwrotnie, T z A). Ta cecha charakterystyczna DNA stoi za niemal całym podejściem sekwencjonowania przez hybrydyzację. Metoda sekwencjonowania przez hybrydyzację w ogólności składa się z dwóch faz: fazy biochemicznej oraz obliczeniowej. W fazie biochemicznej wykorzystuje się tak zwany *chip DNA* specjalnie przygotowany do eksperymentu, aby w procesie dołączania doń sklonowanych jednoniciowych fragmentów DNA możliwe było wykrycie jakie l-mery (oligonukleotydy o pewnej ustalonej długości) tworzą analizowane DNA. Są one później zawarte w tak zwanym *spektrum DNA*.

Chip DNA można sobie wyobrazić jako pewną powierzchnię, na której w wydzielonych sektorach (które dalej będą określane jako sondy) znajdują się oligonukleotydy. Oligonukleotydem nazywamy pewną niewielką jednoniciową sekwencję, zazwyczaj o długości od kilku do kilkunastu nukleotydów. W podejściu klasycznym w jednej sondzie znajduje się jeden, konkretny typ oligonukleotydu. Jest on powielony w wielu kopiach, aby zapewnić odpowiednią siłę sygnału hybrydyzacyjnego.

W czysto klasycznym podejściu chip DNA zawiera wszystkie oligonukleotydy o wyznaczonej długości l. Jeżeli na przykład przyjmiemy, że $l = 10$, wtedy liczba wszystkich możliwych oligonukleotydów wynosi $4^{10} = 1048576$. Tyle też sond na swojej powierzchni musi zawierać wtedy klasyczny chip DNA.

W eksperymencie hybrydyzacyjnym sklonowane kopie jednoniciowego badanego DNA są doprowadzane na powierzchnię chipu. Jeżeli sonda składa się z oligonukleotydów, które są w pełni komplementarne do jakiegoś fragmentu badanego DNA, następuje hybrydyzacja, tj. połączenie jednoniciowego DNA z daną sondą. Po takim eksperymencie biochemicznym sondy, które hybrydyzowały z DNA tworzą tak zwane spektrum DNA. Przyjmuje się, że spektrum zawiera tylko komplementarne do sond podłańcuchy DNA. Dzięki takiemu podejściu można je także potraktować jako zbiór wszystkich podciągów pewnej długości tworzących badane DNA.

W tym miejscu należy zwrócić uwagę na problem błędów hybrydyzacji. Mogą one być

w ogólności dwojakiego rodzaju, z czego jeden z nich dzieli się dalej ze względu na źródło swego pochodzenia. Błędy pozytywne hybrydyzacji to dodatkowe, fałszywe elementy spektrum, czyli podłańcuchy, które w rzeczywistości nie występują w badanym DNA. W wyniku niedoskonałości metody biochemicznej może zajść częściowa hybrydyzacja z sondami, które teoretycznie hybrydyzować nie powinny. Także intensywność sygnału sondy może powodować fałszywe odczyty a co za tym idzie wystąpienie fałszywych nadmiarowych elementów w spektrum. Ten rodzaj błędów jest uzależniony od technologii wykorzystywanej w eksperymencie hybrydyzacyjnym. Poprzez udoskonalanie tego procesu można wpływać na ich minimalizację w spektrum

Drugim rodzajem błędów są błędy negatywne. Jak można się domyślić stanowią one przeciwieństwo błędów pozytywnych - brak pewnych elementów w spektrum DNA, czyli brak pewnych łańcuchów, które w rzeczywistości stanowią część badanego DNA. Ten rodzaj błędów może mieć dwojakie pochodzenie, z czego jedno z nich nie może być niestety nigdy wykluczone. Pierwszym źródłem tych błędów może być, jak poprzednio, niedoskonałość samego technicznego procesu fazy biochemicznej sekwencjonowania przez hybrydyzację. Teoretycznie mógłby być on wyeliminowany, tak jak opisano powyżej, czyli poprzez udoskonalanie technologii wykonywania eksperymentu hybrydyzacyjnego. Niestety, jest też drugie źródło tego rodzaju błędów, wynikające z możliwej budowy samego analizowanego DNA. Jeżeli w badanej sekwencji znajdują się identyczne podłańcuchy o długości l, lecz w różnych miejscach nici DNA, przyłączą się one do tej samej sondy w fazie biochemicznej, jednak precyzyjna informacja ile razy to nastąpiło jest niedostępna. Możliwe jest określenie z pewnym prawdopodobieństwem czy podłańcuch DNA występuje w nim raz czy więcej niż raz, ale generalnie przyjmuje się, że informacja z chipu pozostaje binarna: hybrydyzacja nastąpiła lub nie. Z tego powodu błędy negatywne nie są nigdy całkowicie możliwe do wyeliminowania.

Mając do dyspozycji dane w postaci spektrum DNA można odtworzyć badane DNA używając odpowiednich algorytmów, co ma miejsce w drugiej fazie metody sekwencjonowania przez hybrydyzację - w fazie obliczeniowej. Obecność błędów hybrydyzacji w spektrum sprawia, że sam problem obliczeniowy jest trudny. Kolejnym problemem są rozwiązania niejednoznaczne. Dla instancji odpowiedniego problemu kombinatorycznego, z badanym DNA o długości n oraz l jako długości ciągów w spektrum, może istnieć więcej niż jedno rozwiązanie dopuszczalne. Algorytm zwraca wtedy więcej niż jedną sekwencję DNA, przy czym nie jest możliwe określenie, która z nich jest tą prawidłową, która brała udział w eksperymencie hybrydyzacyjnym fazy biochemicznej SBH.

4.2.2 Izotermiczne sekwencjonowanie przez hybrydyzację

Istnieje wiele odmian metody sekwencjonowania przez hybrydyzację, które w przeciwieństwie do swojej wersji klasycznej nie wymagają używania wszystkich oligonukleotydów o ustalonej wcześniej długości. Jednym z nich jest izotermiczne SBH, gdzie brana jest pod uwagę temperatura dupleksów C/G oraz A/T [6], [8], [5], [21]. Jak wspomniano wcześniej, jednym z głównych powodów złożoności obliczeniowej opisywanej metody są błędy hybrydyzacji. Części z nich nie daje się uniknąć, jak na przykład błędów negatywnych wynikających z powtórzeń, jednak inne mogą być zredukowane poprzez odpowiednie podejścia do procesu hybrydyzacji. Jakość reakcji zachodzących w procesie hybrydyzacyjnym jest zależna od rodzaju nukleotydów budujących podłańcuchy DNA. Wiązania między nukleotydami C/G są ogólnie stabilniejsze niż wiązania A/T, co zostało już wyjaśnione w rozdziale dotyczącym zagadnień biochemicznych związanych z budową DNA. Podejście izotermiczne wykorzystuje oligonukleotydy, w których stała jest nie długość, lecz pewien parametr temperatury topnienia dla całego oligonukleotydu. Biblioteka izotermiczna składa się więc ze wszystkich oligonukleotydów o pewnej temperaturze T, na którą w danym dupleksie składają się sumy stopni wszystkich tworzących go par nukleotydów. Przyjęto model, w którym pary nukleotydów A/T dodają 2 stopnie do łącznej temperatury topnienia dupleksu, pary C/G natomiast dodają 4 stopnie. Bez wchodzenia w szczegóły można przyjąć założenie, że sekwencję DNA zawsze można odtworzyć na podstawie spektrum otrzymanego przy wykorzystaniu dwóch bibliotek izotermicznych różniących się o temperaturą o 2 stopnie Celsjusza. Podejście to okazało skuteczniejsze od klasycznego, szczególnie w przypadku występowania w spektrum błędów negatywnych.

4.2.3 Specyficzny dobór oligonukleotydów na chipie DNA

Innym sposobem podejścia do nieklasycznego SBH jest używanie tylko wybranych oligonukleotydów do bibliotek tworzących chipy DNA [18]. W tym podejściu wybór odpowiednich oligonukleotydów do sond chipu bazuje na wiedzy jaki rodzaj genu czy chromosomu będzie poddawany hybrydyzacji. Dzięki temu możliwe jest opracowaniu relatywnie długich oligonukleotydów (między 10 a 30 par zasad), które następnie znajdą się na powierzchni chipu. W zależności od rodzaju badanego DNA odrzucany jest duży podzbiór oligonukleotydów, co oznacza, że nie będą one brały udziału w procesie hybrydyzacji. Ta metoda sprawdza się bardzo dobrze przy sekwencjonowaniu łańcuchów DNA, co do których istnieje pewna wiedza początkowa o ich budowie. Dzięki temu zapewniona jest duża precyzja procesu hybrydyzacji a co za tym idzie bardzo duża efektywność części obliczeniowej oraz wysoka jakość otrzymywanych rozwiązań - sekwencjonowanych łańcuchów DNA.

4.2.4 Sekwencjonowanie wielofazowe

Kolejnym podejściem do sekwencjonowania przez hybrydyzację są tak zwane protokoły interaktywne funkcjonujące też pod nazwą sekwencjonowania w rundach [22, 33, 49]. Pomysł polega tutaj na przeprowadzaniu hybrydyzacji w kolejnych fazach, gdzie począwszy od drugiej, przygotowanie kolejnej fazy odbywa się na podstawie danych uzyskanych z przeprowadzenia fazy poprzedniej. Chodzi tutaj przede wszystkim o konstrukcję odpowiednich oligonukleotydów w sondach mikromacierzy DNA. W artykule [33] przedstawiającym takie podejście autorzy prezentują wyniki eksperymentów, które dowodzą, że suma sond zaprojektowanych w kolejnych rundach jest znacząco niższa niż ich liczba potrzebna do przeprowadzenia eksperymentu w jednym podejściu klasycznym. Autorzy przedstawiają także algorytmy zaprojektowane do takiego podejścia do SBH oraz wyniki ich wykorzystania. Praca bazuje między innymi na opisie problemu rekonstrukcji superciągu w rundach na podstawie danych uzyskanych z zapytań dotyczących jego podciągów, który to problem został przedstawiony w [48].

Dalszy postęp w tym podejściu został przedstawiony w pracy [22]. Główna teza stawiana przez autorów jest taka, że z wysokim prawdopodobieństwem ciąg DNA o długości s pochodzący ze zbioru n ciągów, może być z sukcesem odtworzony w siedmiu fazach hybrydyzacji, w których w każdej z nich chip użyty do eksperymentu zawierać będzie nie więcej niż $O(n)$ podciągów. Kolejnym rozwinięciem metody jest artykuł z 2003 roku [49], w którym przedstawione jest podejście rekonstruujące dany ciąg poprzez przeprowadzanie kolejnych faz zapytań, bazujących na ogólnym pytaniu "czy podciąg A występuje w (badanym) S?".

Pomysł sekwencjonowania DNA w kolejnych fazach hybrydyzacji przedstawiony został również w pracy [5], w której połączono pomysł wielu rund hybrydyzacji z użyciem izotermicznych bibliotek oligonukleotydów. To połączenie okazało się skuteczniejsze niż obie metody stosowane osobno, przede wszystkim w kontekście problemu, w którym występują błędy hybrydyzacji.

4.2.5 Dodatkowa informacja o lokalizacji

Ciekawa i efektywna odmiana metody SBH przedstawiona została w pracach [2, 24, 56], w których badany był wariant problemu, gdzie z każdym elementem spektrum związana jest przybliżona informacja o jego lokalizacji w oryginalnej badanej sekwencji DNA. W pracy [24] udowodniona została NP-zupełność problemu znajdowania tzw. pozycyjnej ścieżki Eulera w grafie, gdzie położenie każdej krawędzi jest dane jako stały zakres pozycji w całym rozwiązaniu. Przedstawiono również algorytmy rozwiązujące problem sekwencjonowania z wykorzystaniem takiej informacji. W pracy [2] z roku 2001 przedstawiony

został algorytm rozwiązujący problem sekwencjonowania przez hybrydyzację z informacją o położeniu, w formie maksymalnie trzech dozwolonych lokalizacji każdego elementu ścieżki. W ostatnim z przytoczonych na wstępie artykułów ([56]), przedstawiony został model pozycyjnego SBH uwzględniający błędy hybrydyzacji, zarówno pozytywne jak i negatywne.

4.2.6 Sekwencjonowanie równoległe wielu podłańcuchów DNA

W kontekście liczby błędów hybrydyzacji, istnieje zależność między długością DNA n oraz długością używanych w procesie sekwencjonowania oligonukleotydów l, której bezpośrednim skutkiem jest jakość otrzymanego rozwiązania. Możliwe jest jednak 'pocięcie' DNA przed eksperymentem, aby móc efektywnie wykorzystywać do hybrydyzacji oligonukleotydy o niewielkiej długości, np. rzędu 5 par zasad. Istnieje wiele różnych wersji tego podejścia. Ogólna zasada polega na podzieleniu badanego DNA na fragmenty o wielkości rzędu 200-300 par zasad, aby do hybrydyzacji mogły być używane oligonukleotydy o niewielkiej długości. Jedna z najnowszych metod [29] używająca tego podejścia polega na przyczepianiu pociętych fragmentów DNA do pewnej powierzchni, a następnie hybrydyzacji przy użyciu oznakowanych oligonukleotydów (pentamerów w opisywanym przykładzie). Dzięki temu możliwe jest otrzymanie informacji o przyłączeniu danego konkretnego pentameru do każdego unieruchomionego łańcucha DNA. Proces ten jest wykonywany cyklicznie i równolegle na wszystkich fragmentach, jego końcowym efektem jest zestaw spektr, dzięki którym możliwa jest rekonstrukcja każdego z fragmentów DNA, a następnie odtworzenie całej sekwencji.

4.2.7 Sekwencjonowanie z informacją o powtórzeniach

Błędy negatywne mogą wynikać nie tylko z niedoskonałości procesu hybrydyzacji, lecz także z powodu powtórzeń w sekwencji DNA. Tego rodzaju błędów nie można uniknąć w klasycznej wersji metody sekwencjonowania przez hybrydyzację. Prowadzone są jednak badania nad możliwościami wykrywania tego rodzaju błędów [19], [28]. Wpływ dodatkowej wiedzy - o liczbie powtórzeń, jest ciekawym zagadnieniem, które również jest analizowane. W artykule [28] opisane zostały badania, w których założono posiadanie wiedzy o tym, czy dana sonda hybrydyzowała zawsze z tym samym fragmentem badanego DNA, czy z wieloma, mającymi w sekwencji badanej różne położenie. Nawet taka częściowa wiedza o powtórzeniach ma wpływ na jakość rozwiązań zwracanych przez algorytmy dla problemu SBH, co w zacytowanej pracy zostało pokazane na podstawie wyników przeprowadzonych eksperymentów obliczeniowych.

4.2.8 Podejścia algorytmiczne do klasycznej odmiany SBH

Wiele opisanych tutaj modyfikacji oryginalnej metody sekwencjonowania przez hybrydyzację dotyczy zmian w sposobie przeprowadzania eksperymentu hybrydyzacyjnego, np. poprzez dodatkowe eksperymenty, których celem jest otrzymanie nowych danych pomocnych w optymalnej rekonstrukcji analizowanej sekwencji. Istnieje jednak wiele publikacji koncentrujących się na opracowywaniu i badaniu charakterystyk algorytmów odtwarzających DNA ze spektrum. Prace te koncentrują się na różnych wariantach algorytmów radzących sobie z występowaniem w spektrum błędów hybrydyzacji obu typów. Są to algorytmy przybliżone, ponieważ przestrzeń rozwiązań w przypadku problemu SBH z błędami jest zbyt wielka, aby rozważać algorytm dokładny jako realistyczne podejście. Z uwagi na ilość prac tylko kilka zostanie tutaj przytoczonych, prezentujących jednak ciekawe podejścia do obliczeniowej części metody SBH.

W pracy [7] problem sekwencjonowania przez hybrydyzację został określony jako wariant problemu komiwojażera. Zaproponowany został tam algorytm, zdolny radzić sobie z błędami negatywnymi oraz pozytywnymi w spektrum. W artykule zaprezentowano wyniki testów dla bardzo różnych wariantów występowania tych błędów, także w uwzględnieniem błędów negatywnych wynikających z powtórzeń.

Artykuł [55] opisuje algorytm radzący sobie z obydwoma rodzajami błędów. Dla zbadanych sekwencji DNA (pobranych z bazy danych GenBank) zaproponowany algorytm wykazuje znaczną skuteczność w przypadku nawet dużej liczby błędów pozytywnych oraz przy występowaniu (nawet razem z pozytywnymi) podtypu błędów negatywnych wynikających z powtórzeń podsekwencji o długości oligonukleotydów w testowanym DNA.
W artykule [37] autorzy proponują algorytm akceptacji progu (ang. Threshold Accepting, TA) będący modyfikacją podejścia Simulated Annealing, (SA) do rozwiązania problemu asymetrycznego problemu komiwojażera. Zanim jednak algorytm rozpocznie pracę, nad danym spektrum DNA stosowana jest interesująca procedura, mająca na celu usunięciu jak największej części błędów pozytywnych. Elementy idealnego spektrum DNA są ze sobą powiązane w ten sposób, że każdy element poza początkowym i końcowym (sekwencji) nakłada się maksymalnie na długości $l - 1$ z dwoma innymi elementami spektrum. W przypadku błędów negatywnych i pozytywnych jest to zaburzone, jednak możliwe jest testowanie każdego elementu w jakim stopniu jego obecność w spektrum może być "zweryfikowana" poprzez stopień nałożenia z innymi elementami. W ten sposób eliminowane są elementy nakładające się na inne w spektrum w minimalnym przyjętym (odgórnie) stopniu - charakterystyka ta pasuje do błędów pozytywnych, czyli fałszywych elementów nie będących w rzeczywistości częścią badanej sekwencji DNA.

W pracy [17] zaproponowano algorytm genetyczny przy sformułowaniu problemu SBH jako wersji problemu komiwojażera. Algorytm ten, czego dowodzą wyniki prezentowane

w artykule, dobrze radzi sobie ze spektrami DNA obciążonymi obecnością błędów hybrydyzacji. W pracy przedstawiono również pomysł czyszczenia spektrum poprzez redukcję błędów pozytywnych jako elementów, które można zidentyfikować jako niepasujące (w części) do reszty spektrum. Wyniki przedstawiają efektywność podejścia zarówno przy zastosowaniu jak i niestosowaniu tego procesu.

Jeszcze innym podejściem do rozwiązywania problemu SBH jest zastosowanie podejścia nazwanego Ant Colony Optimization [11]. Algorytm ten bazuje na pracy z roku 2004 [15] i jest wykorzystany w procesie odtwarzania sekwencji DNA ze spektrum. Jest to algorytm stochastycznego przeszukiwania, który iteracyjnie w kolejnym kroku buduje rozwiązanie bazujące na tak zwanym modelu feromonowym. Model ten jest zbiorem wartości numerycznych będącym odzwierciedleniem aktualnej wiedzy algorytmu w jaki sposób budować rozwiązania w kolejnych iteracjach. Po każdej z nich wybierane są najlepsze rozwiązania, które następnie służą do uaktualniania modelu. W kolejnych iteracjach jest to powtarzane w celu ulepszania sposobu konstrukcji rozwiązania. Algorytm jest w stanie rekonstruować długie sekwencje DNA w oparciu o spektrum, z dość wysokim współczynnikiem podobieństwa względem sekwencji badanej w danym eksperymencie SBH.

Algorytmy tutaj przedstawione stanowią przykład ciągłego rozwoju metodologii sekwencjonowania w oparciu o podejście SBH. Są to tylko wybrane przykłady pokazujące ciągły rozwój tej metody, nawet w obliczu tak poważnej alternatywy jaką stanowią nowoczesne sekwenatory DNA.

4.2.9 Uniwersalne nukleotydy w metodzie SBH

Zaproponowane w niniejszej rozprawie algorytmy bazują na chipach DNA z pracy [40], w której do konstrukcji oligonukleotydów w sondach tychże chipów użyto idei nukleotydu uniwersalnego. Dokładny opis chipów i konsekwencji ich użycia zostanie szczegółowo przedstawiony w dalszej części pracy. W bieżącym podrozdziale w skrócie zostaną przedstawione prace różnych autorów, bazujące na pomyśle użycia tzw. nukleotydów uniwersalnych do tworzenia sond chipu DNA.

W poprzednich podrozdziałach opisujących różne podejścia do metody SBH stosowana była podstawowa zasada, w myśl której w jednej sondzie chipu znajduje się konkretny typ oligonukleotydu. Liczba wszystkich różnych oligonukleotydów o długości $l = 10$ wymaga ponad miliona sond. W literaturze poświęconej sekwencjonowaniu przez hybrydyzację nie spotyka się prac postulujących zwiększanie tej ilości w podejściu klasycznym do SBH.

Po opublikowaniu klasycznej odmiany SBH, zaczęto zadawać sobie pytanie o możliwości uzyskania oligonukleotydów, w których jeden lub kilka nukleotydów zostałoby zastąpionych innym związkiem chemicznym. Związek taki symulowałby nukleotyd mający właściwość komplementarności do wszystkich nukleotydów naturalnych. W literaturze spotyka się określenie tzw. *nukleotydu uniwersalnego* (ang. universal base). Sonda zawierająca takie oligonukleotydy (tj. mające w sobie nukleotydy uniwersalne) mogłaby hybrydyzować do wielu różnych fragmentów DNA. Znając jednak dokładne właściwości takiego związku można określić zbiór oligonukleotydów komplementarnych do sondy. Innym przypadkiem jest związek chemiczny wykazujący właściwość komplementarności w stosunku do więcej niż jednego naturalnego nukleotydu, lecz nie do wszystkich. W literaturze spotyka się wtedy nazwę *nukleotydu zdegenerowanego* (ang. degenerate base). Od czasu pierwszych pomysłów i propozycji upłynęło już sporo czasu, a nukleotydy, o których mowa istnieją i są używane w różnych badaniach, o czym świadczą publikacje jak na przykład [30], [58], [3] w których zespoły naukowców proponują różne związki chemiczne jako nukleotydy uniwersalne. Kolejne prace warte wymienienia to [57], [23], [50], w których opisywane są właściwości proponowanych związków chemicznych, a także ich wpływ na różne procesy chemiczne, w tym hybrydyzację stosowaną w metodzie SBH.

Właściwości opisanych nukleotydów mogą być symulowane także poprzez umieszczanie w jednej sondzie zbioru różnych oligonukleotydów. Jeśli na przykład przyjmiemy, że w opisie sondy jeden nukleotyd ma być uniwersalny, tj. komplementarny do wszystkich czterech podstawowych, można to osiągnąć także poprzez umieszczenie w sondzie czterech różnych wersji oligonukleotydów. Każda wersja zawiera wtedy zamiast nukleotydu uniwersalnego jeden z nukleotydów naturalnych.

Opisane tu podejścia mają swoje wady i zalety. Umieszczanie różnych wersji oligonukleotydów w jednej sondzie ma tę zaletę nad użyciem nukleotydów uniwersalnych, że są one naturalne przez co ich uzyskanie je łatwiejsze. Nukleotydy uniwersalne są związkami różnymi od nukleotydów standardowych, co ma wpływ na całość reakcji chemicznych w procesie hybrydyzacji. Wadą jednak używania wielu różnych oligonukleotydów w jednej sondzie jest osłabianie siły sygnału sondy po hybrydyzacji. Im więcej różnych oligonukleotydów, tym trudniejsze może być wykrycie hybrydyzacji tylko jednego z nich. Z drugiej strony, do takiej sondy hybrydyzować może (lecz nie musi) więcej różnych podłańcuchów sekwencji DNA. Badania nad różnymi podejściami w sferze zastosowań dla metody SBH są prowadzone, czego przykładem mogą być publikacje [41], [42], [43], [44], [51].

Z punktu widzenia algorytmów opisanych w niniejszej rozprawie, realizacja biochemiczna idei nukleotydu uniwersalnego nie ma znaczenia - czy to poprzez jeden związek chemiczny, czy przez użycie zbioru oligonukleotydów w jednej sondzie. Dane wejściowe dla algorytmów stanowi nieklasyczne spektrum, w którym część lub wszystkie jego elementy traktowane są jako ciągi znaków zbudowane nad pewnym alfabetem. Niektóre znaki tego alfabetu są traktowane jako opisane tutaj nukleotydy różne od naturalnych,

lecz sposób realizacji fazy biochemicznej SBH, która doprowadziła do powstanie spektrum, nie jest dla algorytmu istotny.

Rozdział 5

Gapped Chip

5.1 Wstęp

Treścią niniejszego rozdziału będą problemy związane z sekwencjonowaniem przy użyciu chipu typu Gapped oraz zaproponowany algorytm, rozwiązujący problem sekwencjonowania na bazie spektrum z błędami hybrydyzacji. Jak wspomniano już na wstępie niniejszej rozprawy, zaproponowane w niej algorytmy bazują na konstrukcji chipów DNA, zawartej w artykule z 1994 roku, autorstwa Pavla Pevznera i Roberta Lipshutza [40]. Trzy nieklasyczne projekty budowy chipów DNA służących do sekwencjonowania przez hybrydyzację zostały przedstawione w cytowanym artykule i są to odpowiednio: *Gapped Chip*, *Alternating Chip* oraz *Binary Chip*. Ten i następne dwa rozdziały niniejszej rozprawy poświęcone są badaniu złożoności problemu SBH z udziałem tych chipów, a także dokładnemu opisowi algorytmów do sekwencjonowania opracowanych specjalnie dla tychże chipów. Do tej pory w literaturze naukowej nie pojawiły się algorytmy zdolne praktycznie wykorzystywać spektrum z powyższych chipów.

W niniejszym rozdziale przedstawiony zostanie opis chipu nieklasycznego typu Gapped. Przedstawiona zostanie jego pełna charakterystyka, problemy kombinatoryczne związane z sekwencjonowaniem za jego pomocą oraz zagadnienie ich złożoności obliczeniowej. W ostatnim podrozdziale przedstawiony zostanie opracowany algorytm dla chipu typu Gapped, zdolny radzić sobie z problemem sekwencjonowania DNA, zarówno przy występowaniu błędów negatywnych hybrydyzacji, błędów pozytywnych oraz obu tych typów błędów jednocześnie w spektrum.

5.2 Konstrukcja i charakterystyka chipu typu Gapped

W odniesieniu do podejścia klasycznego, rozpatrywane tutaj podejście nieklasyczne różni się przede wszystkim liczbą oligonukleotydów w jednej sondzie chipu DNA. Jak opisano w poprzednim rozdziale, w podejściu klasycznym sondę tworzy tylko jeden rodzaj oligonukleotydu. W omawianym tutaj podejściu sonda opisywana jest pewnym wzorem, określającym *zbiór* typów oligonukleotydów, który jest w niej zawarty. Gapped Chip będący głównym tematem tego rozdziału, wykorzystuje dodatkowy znak X ponad standardowy 4-literowy alfabet {A, C, G, T} reprezentujący cztery naturalne nukleotydy DNA. Ta litera jest pewnym "symbolem uniwersalnym". Oznacza to, że należy ją traktować jako komplementarną do dowolnego nukleotydu ze zbioru {A, C, G, T}. Innymi słowy reprezentuje ona nukleotyd uniwersalny opisany w poprzednim rozdziale rozprawy.

Chip typu Gapped składa się z dwóch części - klasycznej oraz nieklasycznej. W części klasycznej, wzór opisujący każdą sondę definiuje jeden, konkretny typ oligonukleotydu o zadanej długości. Wzór ten opiera się na standardowym 4-literowym alfabecie {A, C, G, T}. Część druga chipu zawiera sondy, których opis jest tworzony nad alfabetem 5-literowym {A, C, G, T, X}. Podstawowym parametrem określającym chip typu Gapped jest parametr k, który odpowiada za długość oligonukleotydów oraz rozmiar samego chipu. Można powiedzieć, że chip typu Gapped składa się z sond dwóch rodzajów, zdefiniowanych następująco:

$$N_1N_2...N_k \text{ oraz } N_1N_2...N_{k-1}\underbrace{X\ X\ ...\ X}_{k-1}N_k$$

Długość oligonukleotydów w sondach części klasycznej jest równa $l_1 = k$, długość oligonukleotydów zawartych w sondach części nieklasycznej to $l_2 = 2 \cdot k - 1$. Kolejnym parametrem opisującym chip jest jego pojemność (ang. capacity), czyli całkowita liczba sond na jego powierzchni. Pojemność opisywanego chipu to $|C_{gap}(k)| = 2 \cdot 4^k$. Chip składa się więc z dwóch części - klasycznej i nieklasycznej, mających taką samą pojemność sond, równą 4^k.

Wzór sond z części nieklasycznej definiuje pewien zbiór oligonukleotydów. Dla przykładu, jeżeli pewna sonda byłaby opisana wzorem $ACXXG$, w sondzie tej znaleźć się musi (w przypadku nie stosowania nukleotydów uniwersalnych) 16 różnych oligonukleotydów klasycznych, aby 'zasymulować' działanie symbolu X jako uniwersalnego, albo innymi słowy, komplementarnego do każdego nukleotydu z {A, C, G, T}. Te 16 oligonukleotydów z przykładu to:

$$ACAAG\ ;\ ACACG\ ;\ ACAGG\ ;\ ACATG$$
$$ACCAG\ ;\ ACCCG\ ;\ ACCGG\ ;\ ACCTG$$

$$ACGAG\ ;\ ACGCG\ ;\ ACGGG\ ;\ ACGTG$$
$$ACTAG\ ;\ ACTCG\ ;\ ACTGG\ ;\ ACTTG$$

Oczywiście, jak opisane zostało już w rozdziale 4, sonda taka może być zbudowana z użyciem nukleotydów uniwersalnych. Co ważne, jak dokładnie będzie to realizowane w fazie biochemicznej SBH, nie jest istotne z punktu widzenia zaproponowanego algorytmu. Spektrum DNA będące wynikiem eksperymentu hybrydyzacyjnego przeprowadzonego za pomocą chipu typu Gapped, w części składa się z ciągów o długości l_1 będących podciągami badanego DNA, w części zaś z ciągów o długości l_2 będących oznaczeniami zbiorów oligonukleotydów komplementarnych do sond nieklasycznych, które w eksperymencie hybrydyzowały. Ta druga część spektrum, jak zostanie w dalszej części rozdziału opisane, odpowiada za weryfikację każdego kroku algorytmu, który rekonstruuje DNA z elementów części klasycznej. Należy tutaj podkreślić, że o ile sam chip składa się z dwóch części o jednakowej liczbie sond, spektrum DNA zawiera elementy klasyczne i nieklasyczne, których liczba najczęściej jest różna. Wynika to z różnic w długości oligonukleotydów obu połówek chipu, ale przede wszystkim z możliwości wystąpienia błędów hybrydyzacji. Przy teoretycznym braku jakichkolwiek błędów, elementów klasycznych w spektrum jest dokładnie o $k-1$ więcej niż elementów nieklasycznych.

5.3 Zagadnienia prawdopodobieństwa rozgałęzień

W pracy [40] zostały przedstawione podstawy obliczeń, służących do klasyfikacji chipów DNA. W tym podrozdziale zostaną one w całości przytoczone, ponieważ stanowią uzupełnienie tematyki właściwości chipu typu Gapped. Wyliczenia te, co należy podkreślić, zostały przeprowadzone dla przypadku, w którym nie występują błędy hybrydyzacji. Nawet jednak bez obecności błędów, problem bazujący na takim idealnym spektrum może mieć rozwiązania niejednoznaczne. Oznacza to, że nie można określić, która z otrzymanych sekwencji jest w rzeczywistości badanym DNA w metodzie SBH. Z danym chipem DNA związane jest więc pewne prawdopodobieństwo otrzymania rozwiązania jednoznacznego (tzw. *siła rozstrzygania* chipu) oraz *prawdopodobieństwo rozgałęzień*.

Siła rozstrzygania chipu C może być przykładowo określona wzorem:

$$p_n(C) = 1 - \frac{D_n(C)}{4^n} \tag{5.1}$$

Oznaczenie $p_n(C)$ to prawdopodobieństwo odczytania losowej sekwencji o długości n za pomocą chipu C, $D_n(C)$ to liczba wszystkich sekwencji o długości n, które mogą być niejednoznacznie odczytane za pomocą chipu C. Oznacza to, że z takiego chipu mogą powstawać spektra, z których możliwe jest odtworzenie więcej niż jednego rozwiązania.

Niestety, za pomocą wzoru 5.1 trudno jest wyznaczyć wartość siły rozstrzygania, dlatego autorzy pracy [40] proponują w zamian inne podejście.

Wywód zawarty w [40] zostanie tutaj przytoczony w całości, gdyż stanowi on podstawę części testów, którym poddano opracowane algorytmy. Na początku przedstawione zostaną obliczenia odpowiednich wartości charakteryzujących chip klasyczny, następnie obliczenia dla chipu typu Gapped.

Na początku rozpatrzmy sekwencję $F = N_1N_2...N_{m-1}N_mN_{m+1}...N_n$, oraz załóżmy znajomość pierwszych m nukleotydów. Jeżeli przyjmiemy ogólnie, że spektrum sekwencji F otrzymane za pomocą chipu C można zapisać jako:

$$S(C,F) = \{p \in C : \text{sonda p zawiera oligonukleotyd występujący w sekwencji } \overline{F}\} \quad (5.2)$$

gdzie $\overline{F}$ oznacza sekwencję komplementarną do F. Wtedy F_m oznaczać będzie możliwą rekonstrukcję pierwszych m nukleotydów sekwencji F. Tak więc:

$$S(C,F_m) \subset S(C,F) \quad (5.3)$$

Istnieją cztery sposoby rozszerzenia sekwencji F_m o jeden nukleotyd, określone przez F_mA, F_mC, F_mG oraz F_mT. Rozszerzenie sekwencji F_m jest możliwe o nukleotyd N jeżeli:

$$S(C,F_mN) \subset S(C,F) \quad (5.4)$$

Zdefiniujmy teraz:

$$\epsilon(C,F,m) = \begin{cases} 0 & \text{jeżeli warunek (5.4) jest prawdziwy tylko dla jednego z nukleotydów} \\ 1 & \text{w przeciwnym wypadku} \end{cases} \quad (5.5)$$

Sekwencję F nazwiemy jednoznacznie rozszerzalną po m-tym nukleotydzie, jeżeli $\epsilon(C,F,m) = 0$, w przeciwnym wypadku sekwencja F jest niejednoznacznie rozszerzalna. W tym momencie możemy już zdefiniować prawdopodobieństwo rozgałęzień $p(C,n,m)$, jako prawdopodobieństwo rozszerzenia losowej sekwencji o długości n po m–tym nukleotydzie w czasie rekonstrukcji za pomocą chipu C:

$$p(C,n,m) = \frac{1}{4^n}\sum_F \epsilon(C,F,m) \quad (5.6)$$

gdzie suma dotyczy wszystkich 4^n sekwencji F o długości n. Na potrzeby dalszych rozważań przyjmijmy $p(C,n) = p(C,n,m)$. $p(C,n)$ jako funkcja rośnie wraz z n. Dla pewnego ustalonego prawdopodobieństwa rozgałęzień t, maksymalne n spełniające nierówność:

$$p(C,n) \leqslant t \quad (5.7)$$

wyznacza długość sekwencji DNA, którą można jednoznacznie zrekonstruować z danym prawdopodobieństwem rozgałęzień t.

Poniżej przedstawione zostaną dwa schematy obliczeń z [40]. Pierwszy dotyczyć będzie prawdopodobieństwa rozgałęzień dla chipu klasycznego, drugi zaś schemat dotyczyć będzie omawianego prawdopodobieństwa dla chipu typu Gapped.

Rozpatrzmy po raz kolejny sekwencję $F = N_1N_2...N_{m-1}N_mN_{m+1}...N_n$. Dla uproszczenia oznaczmy sekwencję $k-1$ ostatnich nukleotydów podsekwencji $N_1N_2...N_m$ jako V, czyli precyzyjnie: $V = N_{m-k+2}...N_m$. Załóżmy, że $m \geqslant k$ oraz $k \ll n \ll 4^k$, gdzie $4^k = |C(k)|$ jest pojemnością chipu klasycznego dla ustalonej długości oligonukleotydów k. Niech VY_2, VY_3 oraz VY_4 będą możliwymi alternatywnymi rozszerzeniami sekwencji V, w stosunku do znanej sekwencji VN_{m+1}. Sekwencja V wraz z danym, rozszerzającym ją nukleotydem ma wtedy długość k. Załóżmy, że prawdopodobieństwo znalezienia danego podciągu o długości k na danej pozycji w sekwencji F jest równe:

$$\frac{1}{4^k} \tag{5.8}$$

Z drugiej strony prawdopodobieństwo, że taki podciąg nie znajduje się na tej pozycji jest równe:

$$1 - \frac{1}{4^k} \tag{5.9}$$

Prawdopodobieństwo, że taki łańcuch nie znajduje się na żadnej pozycji w F to:

$$(1 - \frac{1}{4^k})^{n-k+1} \tag{5.10}$$

Ponieważ interesuje nas prawdopodobieństwo zdarzenia, że żaden z ciągów VY_2, VY_3, VY_4 nie występuje w F, otrzymujemy:

$$((1 - \frac{1}{4^k})^{n-k+1})^3 \tag{5.11}$$

Teraz możliwe jest określenie prawdopodobieństwa rozgałęzień dla chipu klasycznego oznaczanego dalej jako $C(k)$ oraz sekwencji o długości n, czyli prawdopodobieństwa znalezienia przynajmniej jednego z podciągów VY_2, VY_3, VY_4. Jest ono równe:

$$p(C(k), n) = 1 - ((1 - \frac{1}{4^k})^{n-k+1})^3 \approx \frac{3n}{4^k} = \frac{3n}{|C(k)|} \tag{5.12}$$

Stąd możliwe jest określenie maksymalnej długości sekwencji $n_{max}(C(k), p)$, która może być z danym prawdopodobieństwem jednoznacznie przetworzona przy użyciu chipu:

$$n_{max}(C(k), p) \approx \frac{1}{3} \cdot |C(k)|p \tag{5.13}$$

Na koniec rozpatrzmy parametry dla chipu typu Gapped przedstawione w [40]. Niech $m \geqslant 2k-1$ oraz $n \ll |C_{gap}(k)|$. Niech $U = N_{m-2k+4}...N_{m-k+2}$ oraz $V = N_{m-k+2}...N_m$. Niejednoznaczność występuje tutaj, jeżeli spektrum zawiera zarówno element klasyczny VY_i oraz nieklasyczny $U\underbrace{X\ X\ ...\ X}_{k-1}Y_i$, przy czym $Y_i \neq N_{m+1}$. Prawdopodobieństwa dla elementów klasycznych są takie same jak określono w równaniach (5.8- 5.11). Prawdopodobieństwo, że spektrum nie zawiera elementu (jednego) nieklasycznego opisanego wyżej to w przybliżeniu:

$$(1 - \frac{1}{4^k})^{n-(2k-1)+1} \tag{5.14}$$

Prawdopodobieństwo sytuacji odwrotnej, tj. tego, że spektrum zawiera element nieklasyczny:

$$1 - (1 - \frac{1}{4^k})^{n-(2k-1)+1} \tag{5.15}$$

Wzór określający przybliżone prawdopodobieństwo znalezienia obu elementów - klasycznego i nieklasycznego w sekwencji F to:

$$(1 - (1 - \frac{1}{4^k})^{n-k+1}) \cdot (1 - (1 - \frac{1}{4^k})^{n-2k+2}) \approx \frac{n^2}{4^k \cdot 4^k} \tag{5.16}$$

Zakładając, że $p(C_{gap}(k), n) = P\{VY_i \in S(C_{gap}(k), F)$ oraz $UY_i \in S(C_{gap}(k), F)\}$

$$p(C_{gap}(k), n) \approx 1 - (1 - \frac{n^2}{4^k \cdot 4^k})^3 \approx 3 \cdot \frac{n}{4^k} \cdot \frac{n}{4^k} = \frac{12n^2}{|C_{gap}(k)|^2} \tag{5.17}$$

Stąd można określić maksymalną długość DNA, którą z danym prawdopodobieństwem p można jednoznacznie odczytać za pomocą spektrum $C_{gap}(k)$:

$$n_{max}(C_{gap}(k), p) \approx \frac{1}{\sqrt{12}} \cdot |C_{gap}(k)|\sqrt{p} \tag{5.18}$$

5.4 Problemy oraz ich złożoność obliczeniowa

5.4.1 Podstawowe definicje

Def. 5.4.1. Ciągi typu Gapped
Zbiór S_{G1} zawiera wszystkie elementy zbudowane nad alfabetem {A, C, G, T} o długości $l_1 = k$. Zbiór S_{G2} zawiera wszystkie elementy zbudowane nad alfabetem {A, C, G, T, X} o długości $l_2 = 2k - 1$ spełniające warunki:

1) litery A, C, G lub T alfabetu występują tylko na pozycjach od 1 do $k - 1$ oraz na pozycji $2k - 1$;

2) litera X alfabetu występuje tylko na pozycjach od k do $2k - 2$.

Ciągi typu Gapped są elementami zbioru $S_{G1} \cup S_{G2}$

Def. 5.4.2. Nakładanie ciągów typu Gapped
Ciągi p o długości l_p oraz q o długości l_q zbudowane nad alfabetem {A, C, G, T, X} nakładają się na pewnej długości $r \leqslant min(l_p, l_q)$ jeżeli dla dwóch ich podciągów p_{a+i} oraz q_{b+i} gdzie a i b to indeks pierwszego znaku podciągu w nałożeniu, odpowiednio dla p i q, $i = 0, 1, ..., r - 1$, spełnione są warunki:

1) jeżeli na pozycji i w pierwszym podciągu znajduje się litera z {A, C, G, T}, na tej samej pozycji w drugim podciągu znajduje się identyczna litera z tego alfabetu, czyli $p_{a+i} = q_{b+i}$;

2) jeżeli na pozycji i w jednym podciągu znajduje się litera X, na pozycji i w drugim podciągu może znajdować się dowolna litera z {A, C, G, T, X};

3) jeden z indeksów: a lub b musi być równy 1 co oznacza, że jeden z podciągów p_{a+i} lub q_{b+i} utworzony jest z początkowych r liter ciągu, od którego pochodzi.

Efektem nałożenia się p i q jest superciąg t ciągów p i q, w którym każdy znak $t_{a+b+i-1}$ (tj. znak części wspólnej p i q w t) stworzony jest następująco:

1) jeżeli $p_{a+i} = q_{b+i}$, wtedy $t_{a+b+i-1} = p_{a+i} = q_{b+i}$;

2) jeżeli $p_{a+i} = X$ oraz $q_{b+i} \in \{A, C, G, T\}$, wtedy $t_{a+b+i-1} = q_{b+i}$;

3) jeżeli $q_{b+i} = X$ oraz $p_{a+i} \in \{A, C, G, T\}$, wtedy $t_{a+b+i-1} = p_{a+i}$.

Def. 5.4.3. Uogólniony superciąg typu Gapped
Dla danego zbioru S_G ciągów typu Gapped, uogólnionym superciągiem o długości n typu Gapped jest taki superciąg, który składa się ze wszystkich elementów zbioru S_G nakładających się zgodnie z definicją nakładania ciągów typu Gapped.

Poniższe definicje odnoszą się do klasycznej wersji metody SBH. Będą one dalej modyfikowane w celu dostosowania do odpowiednich chipów i problemów, w niniejszym rozdziale oraz w dwóch kolejnych, dotyczących chipów typu Alternating i Binary.

Def. 5.4.4. Spektrum DNA
Przez spektrum DNA definiujemy zbiór S ciągów o długości l pochodzących z fazy biochemicznej metody SBH, gdzie badano sekwencję DNA o długości n.

Def. 5.4.5. Idealne spektrum DNA
Idealne spektrum DNA S^{is} składa się ze wszystkich ciągów o długościach l bez powtórzeń, pochodzących z sekwencji DNA o długości n.

Def. 5.4.6. Idealne multispektrum DNA
Idealne multispektrum DNA S^{im} składa się ze wszystkich ciągów o długościach l, pochodzących z sekwencji DNA o długości n. Każdy element idealnego multispektrum powtarza się tyle razy, ile odpowiadający mu podciąg w sekwencji DNA. Liczność idealnego multispektrum dana jest wzorem $|S^{im}| = n - l + 1$.

5.4.2 Gapped SBH bez błędów hybrydyzacji

Def. 5.4.7. Idealne spektrum problemu Gapped SBH(GSBH)
Idealne spektrum składa się ze wszystkich ciągów typu Gapped bez powtórzeń, o długościach $l_1 = k$ oraz $l_2 = 2k - 1$ sekwencji DNA o długości n. W przypadku spektrum idealnego moce zbiorów $S^{(is)}_{G1}$ oraz $S^{(is)}_{G2}$ spełniają nierówności $|S^{(is)}_{G1}| \leqslant n - l_1 + 1$ oraz $|S^{(is)}_{G2}| \leqslant n - l_2 + 1$.
Wszystkie ciągi ze zbioru $S^{(is)}_{G2}$ (zbudowane nad alfabetem 5-literowym) nakładają się całkowicie na ciągi o długości l_2 z badanej sekwencji DNA zgodnie z definicją nakładania się ciągów typu Gapped.

Problem 5.4.1. Problem Gapped SBH bez błędów w wersji decyzyjnej (GSBH-efd, error-free, decision)
Instancja: zbiór $S_G = S_{G1} \cup S_{G2}$ taki, że $S_{G1} = S^{(is)}_{G1}$ oraz $S_{G2} = S^{(is)}_{G2}$, będący idealnym spektrum GSBH, długość n sekwencji DNA, $|S_{G1}| = n - l_1 + 1$ oraz $|S_{G2}| = n - l_2 + 1$.
Odpowiedź: TAK jeżeli istnieje uogólniony superciąg typu Gapped o długości n zawierający wszystkie elementy zbiorów S_{G1} oraz S_{G2} zgodnie z definicją nakładania ciągów typu Gapped, składający się tylko z liter z alfabetu {A, C, G, T}, NIE w przeciwnym wypadku.

W tym wypadku problem decyzyjny jest łatwy obliczeniowo - wiadomo, że istnieje jego rozwiązanie, zważywszy na założenie, że instancja składa się ze spektrum idealnego czyli zawierającego wszystkie fragmenty / podciągi badanego DNA. Oznacza to, że zbiór wszystkich instancji problemu $D_{\Pi_{GSBH-efd}}$ jest równy zbiorowi wszystkich instancji problemu, dla których odpowiedź brzmi TAK: $D_{\Pi_{GSBH-efd}} = Y_{\Pi_{GSBH-efd}}$.

Problem 5.4.2. Problem Gapped SBH bez błędów w wersji przeszukiwania (GSBH-efs, error-free, search)
Instancja: zbiór $S_G = S_{G1} \cup S_{G2}$ taki, że $S_{G1} = S_{G1}^{(is)}$ oraz $S_{G2} = S_{G2}^{(is)}$, będący idealnym spektrum GSBH, długość n sekwencji DNA, $|S_{G1}| = n - l_1 + 1$ oraz $|S_{G2}| = n - l_2 + 1$.
Odpowiedź: uogólniony superciąg typu Gapped o długości n zawierający wszystkie elementy zbiorów S_{G1} oraz S_{G2} zgodnie z definicją nakładania ciągów typu Gapped, składający się tylko liter z alfabetu {A, C, G, T}.

5.4.3 Gapped SBH z błędami negatywnymi

Błędy negatywne mają dwa źródła: brak pewnych elementów ze spektrum idealnego lub powtarzanie się pewnych fragmentów w DNA mających długość elementów użytych do utworzenia spektrum. Aby uwzględnić także ten drugi rodzaj błędów, konieczna jest definicja idealnego multispektrum:

Def. 5.4.8. Idealne multispektrum problemu Gapped SBH(GSBH)
Idealne multispektrum składa się ze wszystkich ciągów typu Gapped o długościach $l_1 = k$ oraz $l_2 = 2k - 1$ sekwencji DNA o długości n. W przypadku multispektrum idealnego moce multizbiorów $S_{G1}^{(im)}$ oraz $S_{G2}^{(im)}$ są równe odpowiednio: $|S_{G1}^{(im)}| = n - l_1 + 1$ oraz $|S_{G2}^{(im)}| = n - l_2 + 1$. Każdy element w multispektrum powtarza się tyle razy ile odpowiadający mu podciąg w DNA.
Każdy ciąg e_{G2} ze zbioru $S_{G2}^{(im)}$ (zbudowany nad alfabetem 5-literowym) powtarza się w spektrum tyle razy, ile razy powtarza się w sekwencji podciąg o długości l_2, na który ciąg e_{G2} nakłada się zgodnie z definicją nakładania dla ciągów typu Gapped.

Problem 5.4.3. Problem Gapped SBH z błędami negatywnymi w wersji decyzyjnej (GSBH-ned, negative errors, decision)
Instancja: zbiór $S_G = S_{G1} \cup S_{G2}$ taki, że $S_{G1} \subset S_{G1}^{(im)}$ oraz $S_{G2} \subset S_{G2}^{(im)}$, będący podzbiorem bez powtórzeń idealnego multispektrum GSBH, długość n sekwencji DNA.
Odpowiedź: TAK jeżeli istnieje uogólniony superciąg zbudowany tylko nad alfabetem {A, C, G, T} o długości mniejszej lub równej n zawierający wszystkie elementy zbiorów S_{G1} oraz S_{G2} zgodnie z definicją nakładania ciągów typu Gapped.

Zbiór wszystkich instancji powyższego problemu *nie* jest równy zbiorowi instancji, dla których odpowiedź brzmi TAK. Spektrum problemu jest podzbiorem idealnego multispektrum pochodzącego z podzielenia sekwencji DNA na mniejsze podciągi, jednakże charakter poszukiwanego rozwiązania sprawia, że istnieją takie instancje, które nie mają rozwiązania (tj. odnaleziony superciąg o zadanej długości $\leqslant n$ nie będzie zbudowany tylko nad alfabetem {A, C, G, T} lecz nad {A, C, G, T, X}). Przykładem jest na przykład instancja ze znaczącą przewagą elementów drugiego ze zbiorów - S_{G2}. W takim wypadku może nie być możliwe odtworzenie uogólnionego superciągu składającego się według definicji z liter ze zbioru {A, C, G, T}, nawet jeśli jest ona podzbiorem zbioru instancji idealnego multispektrum. Zbiór wszystkich instancji problemu GSBH-ned jest równy sumie zbiorów odpowiednio wszystkich instancji z odpowiedzią TAK i wszystkich instancji z odpowiedzią NIE: $D_{\Pi_{GSBH-ned}} = Y_{\Pi_{GSBH-ned}} \cup N_{\Pi_{GSBH-ned}}$

Problem 5.4.4. Problem Gapped SBH z błędami negatywnymi w wersji przeszukiwania, (GSBH-nes, negative errors, search)
Instancja: zbiór $S_G = S_{G1} \cup S_{G2}$ taki, że $S_{G1} \subset S_{G1}^{(im)}$ oraz $S_{G2} \subset S_{G2}^{(im)}$, będący podzbiorem bez powtórzeń idealnego multispektrum GSBH, długość n sekwencji DNA.
Odpowiedź: uogólniony superciąg typu Gapped zbudowany tylko nad alfabetem {A, C, G, T} o długości mniejszej lub równej n zawierający wszystkie elementy zbiorów S_{G1} oraz S_{G2} zgodnie z definicją nakładania ciągów typu Gapped.

Twierdzenie 5.4.1. Problem GSBH-nes jest silnie NP-trudny.

Dowód
Zbiór wszystkich instancji problemu GSBH-nes jest równy zbiorowi wszystkich instancji problemu decyzyjnego GSBH-ned, tj. $D_{\Pi_{GSBH-nes}} = D_{\Pi_{GSBH-ned}}$. Należy zauważyć, że szczególnym przypadkiem problemu GSBH-nes jest problem klasycznego sekwencjonowania metodą SBH z błędami negatywnymi, czyli problem, którego silna NP-trudność została udowodniona w pracy [10]. Jeśli wyobrazimy sobie podproblem problemu GSBH-nes, w którym błędy negatywne powodują brak wszystkich elementów zbioru S_{G2}, otrzymujemy instancję problemu SBH z błędami negatywnymi. Taki szczególny przypadek wciąż tworzy pełnoprawną instancję ze zbioru instancji problemu GSBH-nes. Co dość oczywiste, "uogólniony superciąg" będący rozwiązaniem zawsze składać się będzie z liter alfabetu {A, C, G, T}, ponieważ w takim szczególnym przypadku znak X w spektrum nie występuje.
Zbiór wszystkich takich instancji problemu GSBH-nes, w których nie występują w ogóle ciągi nieklasyczne (tj. S_{G2} jest wtedy zbiorem pustym), stanowi zbiór wszystkich instancji problemu klasycznego SBH z błędami negatywnymi w wersji przeszukiwania. Ponieważ jest to problem silnie NP-trudny, a jednocześnie jest on podproblemem GSBH-nes, ten ostatni jest również problemem silnie NP-trudnym. □

5.4.4 Gapped SBH z błędami pozytywnymi

Definicje problemów sformułowane poniżej wykorzystują definicję idealnego spektrum GSBH 5.4.7 oraz idealnego multispektrum dla GSBH 5.4.8.

Problem 5.4.5. Problem GSBH z błędami pozytywnymi w wersji decyzyjnej (GSBH-ped, positive errors, decision)
Instancja: zbiór $S_G = S_{G1} \cup S_{G2}$ taki, że $S_{G1}^{(is)} = S_{G1}^{(im)} \subset S_{G1}$ oraz $S_{G2}^{(is)} = S_{G2}^{(im)} \subset S_{G2}$, zawierające w sobie idealne spektrum GSBH, długość n sekwencji DNA.
Odpowiedź: TAK jeżeli istnieje uogólniony superciąg typu Gapped zbudowany tylko nad alfabetem {A, C, G, T} o długości n zawierający $|S_{G1}^{(is)}|$ różnych elementów ze zbioru S_{G1} oraz $|S_{G2}^{(is)}|$ różnych elementów ze zbioru S_{G2} zgodnie z definicją nakładania ciągów typu Gapped.

Zbiór wszystkich instancji powyższego problemu jest równy zbiorowi instancji, dla których odpowiedź brzmi TAK: $D_{\Pi_{GSBH-ped}} = Y_{\Pi_{GSBH-ped}}$. Idealne spektrum dla GSBH jest tutaj także podzbiorem spektrum problemu powyższego. Wiadomo, że w idealnym spektrum GSBH istnieje uogólniony superciąg o długości n zbudowany tylko nad alfabetem {A, C, G, T}, tak więc złożoność powyższego problemu decyzyjnego jest wielomianowa: odpowiedź zawsze brzmi TAK.

Problem 5.4.6. Problem GSBH z błędami pozytywnymi w wersji przeszukiwania, (GSBH-pes, positive errors, search)
Instancja: zbiór $S_G = S_{G1} \cup S_{G2}$ taki, że $S_{G1}^{(is)} = S_{G1}^{(im)} \subset S_{G1}$ oraz $S_{G2}^{(is)} = S_{G2}^{(im)} \subset S_{G2}$, zawierające w sobie idealne spektrum GSBH, długość n sekwencji DNA.
Odpowiedź: uogólniony superciąg typu Gapped zbudowany tylko nad alfabetem {A, C, G, T} o długości n zawierający $|S_{G1}^{(is)}|$ różnych elementów ze zbioru S_{G1} oraz $|S_{G2}^{(is)}|$ różnych elementów ze zbioru S_{G2} zgodnie z definicją nakładania ciągów typu Gapped.

Zbiór wszystkich instancji powyższego problemu jest równy zbiorowi instancji problemu decyzyjnego $D_{\Pi_{GSBG-pes}} = D_{\Pi_{GSBH-ped}}$. Należy dodać, że jest on przez to równy zbiorowi instancji z odpowiedzią TAK dla problemu decyzyjnego
: $D_{\Pi_{GSBG-pes}} = Y_{\Pi_{GSBH-ped}}$.

Problem 5.4.7. Problem quasi-pozytywnego GSBH w wersji decyzyjnej (qGSBH-ped, quasi, positive errors, decision)
Instancja: zbiór $S_G = S_{G1} \cup S_{G2}$ zawierający ciągi typu Gapped o długościach l_1 i l_2, długość n badanej sekwencji DNA.
Odpowiedź: TAK jeżeli istnieje uogólniony superciąg typu Gapped zbudowany tylko nad alfabetem {A, C, G, T} o długości n zawierający $n - l_1 + 1$ różnych elementów ze zbioru

S_{G_1} oraz $n-l_2+1$ różnych elementów ze zbioru S_{G2} zgodnie z regułą nakładania ciągów typu Gapped.

Zbiór wszystkich instancji z odpowiedzią TAK problemu qGSBH-ped jest równy zbiorowi wszystkich instancji z odpowiedzią TAK dla problemu GSBH-ped. Zbiór ogólnie wszystkich instancji GSBH-ped jest jednak podzbiorem zbioru wszystkich instancji problemu qGSBH-ped, ponieważ ten drugi zawiera także instancje, dla których odpowiedź brzmi NIE, $Y_{\Pi_{qGSBH-ped}} = Y_{\Pi_{GSBH-ped}}$, $D_{\Pi_{GSBH-ped}} \subset D_{\Pi_{qGSBH-ped}}$

Twierdzenie 5.4.2. Problem qGSBH-ped jest silnie NP-zupełny.

Dowód

W tej części podrozdziału przedstawiony zostanie dowód silnej NP-zupełności problemu qGSBH-ped. Punktem wyjścia zaproponowanego podejścia jest transformacja problemów klasycznego SBH z artykułu [10]. W artykule tym omówiona została złożoność obliczeniowa problemu SBH z błędami pozytywnymi w wersji przeszukiwania. Dokonano tego pośrednio, poprzez transformację od problemu skierowanej ścieżki Hamiltona między dwoma wierzchołkami (Directed Hamiltonian Path Between Two Vertices, DHPBTW) do problemu pozytywnego quasi-sekwencjonowania. Podobne podejście zostanie zastosowane w niniejszej rozprawie.

Problem 5.4.8. Skierowana ścieżka Hamiltona między dwoma wierzchołkami, DHPBTW
Instancja: graf skierowany $H = (V, A)$, z zaznaczonym wierzchołkiem początkowym i końcowym s i t.
Odpowiedź: TAK, jeżeli istnieje ścieżka Hamiltona pomiędzy s i t.

Poniżej opisana zostanie zmodyfikowana transformacja z [10]. Modyfikacja dodaje dodatkowy znak T na końcu każdego odpowiednika wierzchołka grafu. Zmiana ta powoduje dalsze zmiany w budowie kolejnych ciągów, będących odpowiednikami struktury grafu z instancji problemu DHPBTW. Przebiega ona następująco:

Mając daną instancję DHPBTV postępujemy następująco :

1) każdemu wierzchołkowi $v \in V$ przydzielamy unikalną nazwę $e(v)$ o długości $\lceil log_2|V| \rceil$ nad alfabetem {A, C};

2) definiujemy $l = 2\lceil log_2|V| \rceil + 3$ jako długość wszystkich elementów w S;

3) budujemy zbiór S tak, że dla każdego wierzchołka $v \in V$ budujemy jeden oligonukleotyd w formie $e(v) \cdot G \cdot e(v) \cdot TT$ ($\cdot$ oznacza konkatenację, G oraz T to nukleotydy);

4) do S dodajemy $l-1$ oligonukleotydów dla każdego łuku $(u, v) \in A$, oligonukleotydy w formie $u_2 \cdot u_3 \cdot \cdots \cdot u_l v_1, u_3 \cdot u_4 \cdot \cdots \cdot v_1 \cdot v_2, \ldots, u_l \cdot v_1 \cdot \cdots \cdot v_{l-2} \cdot v_{l-1}$ (gdzie $u_1 \cdot u_2 \cdot \cdots \cdot u_l$ to oligonukleotyd oznaczający wierzchołek u). Każdy podzbiór elementów będących odpowiednikiem łuku będzie nazywany ścieżką łukową;

5) do S dodajemy l oligonukleotydów startowych w formie $t_1 \cdot g_2 \cdot \cdots \cdot g_{l-1}, g_2 \cdot g_3 \cdot \cdots \cdot g_l \cdot s_1, \ldots, g_l \cdot s_1 \cdot \cdots \cdot s_{l-2} \cdot s_{l-1}$ (gdzie $t_1 = T, g_i = G, 2 \leqslant i \leqslant l$ oraz $s_1 \cdot s_2 \cdot \cdots \cdot s_l$ to oligonukleotyd reprezentujący wierzchołek startowy s);

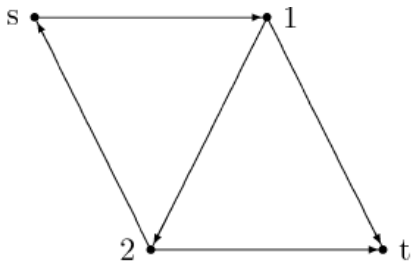
6) do S dodajemy l oligonukleotydów końcowych w formie $t_2 \cdot t_3 \cdot \cdots \cdot t_l \cdot k_1, t_3 \cdot t_4 \cdot \cdots \cdot k_1 \cdot k_2, \ldots, t_l \cdot k_1 \cdot \cdots \cdot k_{l-2} \cdot k_{l-1}, k_1 \cdot k_2 \cdot \cdots \cdot k_l, k_2 \cdot k_3 \cdot \cdots \cdot k_{l-1} \cdot G, k_3 \cdot k_4 \cdot \cdots \cdot k_{l-2} \cdot G \cdot T$ (gdzie $k_i = T, 1 \leqslant i \leqslant l-1$ oraz $t_1 \cdot t_2 \cdot \cdots \cdot t_l$ to oligonukleotyd reprezentujący wierzchołek końcowy t).

Powyższa transformacja umożliwia utworzenie instancji problemu SBH z błędami pozytywnymi. Zbiór będący spektrum instancji problemu SBH z błędami powstaje poprzez włożenie doń tylko nie powtarzających się łańcuchów powstałych po transformacji. Powtórzenia mogą wystąpić tylko wtedy, jeśli z danego wierzchołka wychodzą łuki do dwóch lub więcej wierzchołków, lub więcej niż jeden łuk wchodzi do jednego wierzchołka. Aby udowodnić prawidłowość transformacji zostało w [10] wykazane, że wspomniana wcześniej ścieżka Hamiltona istnieje wtedy i tylko wtedy, gdy istnieje sekwencja utworzona z elementów zbioru S, o długości $n = l(|V| + 2)$, zawierająca $l(|V| + 1) + 1$ różnych elementów. Oczywiście jak już napisano wcześniej, oryginalna transformacja operuje na ciągach, w których odpowiedniki wierzchołków mają jedną literę T na końcu. Ta zmiana nie ma wpływu na liczbę elementów zbioru potrzebnych do uzyskania prawidłowego rozwiązania. Długość potrzebnego superciągu zwiększa się, lecz wzór na nią pozostaje niezmieniony ($n = l(|V| + 2)$). Zmiana w transformacji, o czym będzie jeszcze mowa, ma na celu zapewnie, że tak jak w oryginalnej transformacji, w rozwiązaniu problemu nie powtarza się żaden podciąg o długości l, tak po modyfikacji, nie powtarza się w nim również żaden podciąg o długości $l - 1$.

Aby przejść od problemu DHPBTW do problemu quasi-Gapped, należy wprowadzić dalsze kroki transformacji. W tym miejscu zostanie rozpisany przykład dla grafu danego na rysunku 5.1:

Po przeprowadzeniu kroków zmodyfikowanej transformacji dla danego grafu, powstaną następujące ciągi (elementy zbioru S):

1) Odpowiedniki wierzchołków (kolejno s 1 2 t:)
   $AAGAATT$, $ACGACTT$, $CAGCATT$, $CCGCCTT$



Rysunek 5.1: Graf z instancji problemu DHPBTW [10]

2) Elementy prowadzące do wierzchołka s:
$TGGGGGG$, $GGGGGGA$, $GGGGGAA$, $GGGGAAG$, $GGGAAGA$, $GGAAGAA$, $GAAGAAT$

3) Elementy wychodzące z wierzchołka t:
$CGCCTTT$, $GCCTTTT$, $CCTTTTT$, $CTTTTTT$, $TTTTTTT$, $TTTTTTG$, $TTTTTGT$

4) Elementy tworzące łuki grafu (ścieżki łukowe):
(s,1) - $AGAATTA$, $GAATTAC$, $AATTACG$, $ATTACGA$, $TTACGAC$, $TACGACT$
(1,2) - $CGACTTC$, $GACTTCA$, $ACTTCAG$, $CTTCAGC$, $TTCAGCA$, $TCAGCAT$
(1,t) - $CGACTTC$, $GACTTCC$, $ACTTCCG$, $CTTCCGC$, $TTCCGCC$, $TCCGCCT$
(2,s) - $AGCATTA$, $GCATTAA$, $CATTAAG$, $ATTAAGA$, $TTAAGAA$, $TAAGAAT$
(2,t) - $AGCATTC$, $GCATTCC$, $CATTCCG$, $ATTCCGC$, $TTCCGCC$, $TCCGCCT$

Jak napisano już w samym opisie transformacji, elementy będące odpowiednikami łuków w grafie ($l-1$ elementów na łuk) nazywane będą dalej "ścieżkami łukowymi". Także elementy prowadzące do wierzchołka s oraz wychodzące z t będą dalej określane tym mianem.

Do tego momentu utworzono zbiór S, który po usunięciu powtórzeń tworzy zbiór S_{G1} instancji problemu qGSBH-ned. Aby utworzyć zbiór S_{G2} z elementami o długości $2l-1$, należy sprawdzić, dokąd prowadzą dostępne ścieżki łukowe od każdego elementu zbioru S:

1. Każdy element zbioru S skracamy do długości $l-1$, biorąc pod uwagę tylko pierwsze $l-1$ liter. Tworzymy w ten sposób pierwsze $l-1$ znaków elementu nieklasycznego. Kolejne $l-1$ znaków to znaki X.

2. Sprawdzamy, do jakich elementów prowadzą dostępne ścieżki łukowe (rozpisane już w zbiorze S) $l-1$ kroków dalej. Krokiem jest nakładanie na $l-1$ znakach kolejnych ciągów na ścieżce łukowej. Ostatnia litera elementu osiągniętego po wyznaczonej liczbie kroków (l-tego licząc od elementu rozpatrywanego w punkcie 1), stanowi ostatnią literę elementu nieklasycznego.

3. Jeśli, idąc po kolejnych ciągach ścieżki łukowej, natrafiamy na element będący odpowiednikiem wierzchołka w grafie G, musimy rozpatrzyć wszystkie możliwe ścieżki łukowe w S, które od niego prowadzą. Jeśli prowadzą one do ciągów kończących się na różne litery, wtedy powstanie odpowiednio więcej elementów nieklasycznych, mających te same pierwsze $l-1$ znaków (punkt 1)). Jeśli różne ścieżki łukowe prowadzą do utworzenia tych samych elementów, wtedy w spektrum umieszczany jest tylko jeden.

Na przykład, dla elementu $AAGAATT$ (odpowiednik wierzchołka s), jedyna ścieżka łukowa, którą można pójść to (s,1). Dokładnie ($l-1=6$) kroków dalej znajduje się jej ostatni element: $TACGACT$. Jego ostatnia litera, a także 6 pierwszych liter odpowiednika wierzchołka s, od którego zaczęto szukanie ścieżki w tym przykładzie, tworzy element nieklasyczny, z literami X na pozostałych pozycjach: $AAGAAT\ XXXXXX\ T$.

Przykład 2: trzeci element ścieżki łukowej (s,1) to $AATTACG$. Idąc od niego dochodzimy w 4 kroku do elementu będącego odpowiednikiem wierzchołka "1" w grafie ($ACGACTT$, ponieważ do niego prowadzi ścieżka (s,1)). W tym miejscu musimy przejść jeszcze 2 kroki. Z elementu $ACGACTT$ prowadzą dwie ścieżki łukowe - (1,2) oraz (1,t), ponieważ jest on odpowiednikiem wierzchołka s, z którego prowadzą dwa odpowiadające im łuki. W obu tych ścieżkach łukowych musimy pobrać ostatnią literę ich drugich elementów (drugi element ścieżki (1,2) to $GACTTCA$, natomiast w (1,t) to element $GACTTCC$). Licząc od ciągu, od którego rozpoczęto przemarsz ścieżkami łukowymi, obydwa poszukiwane ciągi znajdują się 6 kroków dalej od ciągu startowego: 3 ciągi następujące po nim na ścieżce łukowej (s,1), jeden ciąg do którego ścieżka ta prowadzi (odpowiednik wierzchołka "1"), pierwsze ciągi ścieżek (1,2) i (1,t) to krok 5. Drugie elementy tych ścieżek, w ostatnim kroku (6), to te, o których mowa w przykładzie - z nich pobieramy ostatni znak do budowy elementu nieklasycznego. Ostatnie ich znaki to A oraz T, tak więc z elementu $AATTACG$, od którego zaczęto w tym przykładzie przemarsz, powstaną dwa elementy nieklasyczne: $AATTAC\ XXXXXX\ A$ oraz $AATTAC\ XXXXXX\ T$. Obydwa elementy są umieszczane w zbiorze S_{G2} o ile jeszcze w nim nie występują (co jednak nie może się zdarzyć, jeżeli w danej chwili rozpatrywana jest ścieżka będąca częścią poprawnego rozwiązania - o tym będzie mowa dalej).

Należy zauważyć, że dla zbioru elementów końcowych ze zbioru S (są to elementy wychodzące z wierzchołka t:

$CGCCTTT\ GCCTTTT\ CCTTTTT\ CTTTTTT\ TTTTTTT\ TTTTTTG\ TTTTTGT$), element nieklasyczny jest możliwy do utworzenia tylko z elementu $CGCCTTT$ i jest to zawsze $CGCCTT\ XXXXXX\ T$. Tym elementem musi się też kończyć superciąg wynikowy (zgodnie z regułami nakładania). Poniżej rozpisane elementy nieklasyczne wg. powyższej transformacji (element x (małe) oznacza w rzeczywistości ciąg znaków $XXXXXX$, oznaczenie $ACTTCAxA/C$ to uproszczony zapis dwóch elementów: $ACTTCAxA$ oraz $ACTTCAxC$, ponieważ różnią się ostatnim znakiem, obydwa znajdą się w spektrum)

1) Dla zbioru ciągów początkowych:
   $TGGGGGxT$, $GGGGGGxT$, $GGGGGAxA$, $GGGGAAxC$, $GGGAAGxG$, $GGAAGAxA$, $GAAGAAxC$

2) Dla odpowiednika wierzchołka s w zbiorze S:
   $AAGAATxT$

3) Dla ścieżki (s,1):
   $AGAATTxT$, $GAATTAxC/C$, $AATTACxA/C\ ATTACGxG/G$, $TTACGAxC/C$, $TACGACxA/C$

4) Dla odpowiednika wierzchołka 1 w zbiorze S: $ACGACTxT/T$

5) Dla ścieżki (1,2):
   $CGACTTxT$, $GACTTCxA/C$, $ACTTCAxA/C$, $CTTCAGxG/G$, $TTCAGCxA/C$, $TCAGCAxA/C$

6) Dla odpowiednika wierzchołka 2 w zbiorze S:
   $CAGCATxT/T$

7) Dla ścieżki (1,t):
   $CGACTTxT$, $GACTTCxT$, $ACTTCCxT$, $CTTCCGxT$, $TTCCGCxT$, $TCCGCCxT$

8) Dla odpowiednika wierzchołka t w zbiorze S:
   $CCGCCTxG$

9) Dla ścieżki (2,s):
   $AGCATTAxT$, $GCATTAxA$, $CATTAAxC$, $ATTAAGxG$, $TTAAGAxA$, $TAAGAAxC$

10) Dla ścieżki (2,t):
    $AGCATTxT$, $GCATTCxT$, $CATTCCxT$, $ATTCCGxT$, $TTCCGCxT$, $TCCGCCxT$

11) Odpowiedniki elementów kończących superciąg ze zbioru S to: $CGCCTTT$, $GCCTTTT$, $CCTTTTT$, $CTTTTTT$, $TTTTTTT$, $TTTTTTG$, $TTTTTGT$. Tylko pierwszy element - $CGCCTTT$ pomoże utworzyć element nieklasyczny: $CGCCTT\ XXXXXX\ T$. Dla pozostałych wymienionych w tym podpunkcie elementów nie jest to możliwe, ani do czegokolwiek wymagane. Wynika to stąd, że jeden element nieklasyczny tutaj utworzony nakładać się będzie na sam koniec superciągu będącego rozwiązaniem.

Elementy będące błędami pozytywnymi tworzone są tylko, jeśli z danego wierzchołka grafu G wychodzi więcej niż jeden łuk. Jeśli chodzi o prawidłowe rozwiązanie, ma ono usystematyzowaną postać: ciąg $TGGGGGG$ na początku, za nim odpowiednik wierzchołka s: $AAGAATT$, $TTTTTGT$ na końcu a przed nim odpowiednik wierzchołka t: $CCGCCTT$. W środku obok siebie znajdują się odpowiedniki (także 7-literowe w przykładzie) wierzchołków grafu, w kolejności, która tworzy w tym grafie ścieżkę Hamiltona pomiędzy wierzchołkami s i t. Przy użyciu pokazanej transformacji umożliwiającej konstrukcję zbioru S_{G1}, nie jest możliwe powtórzenie się elementów nieklasycznych w superciągu będącym odpowiednikiem ścieżki Hamiltona. Przy takiej budowie elementów zbioru S_{G1} o długości l, żaden podciąg $l-1$ literowy w rozwiązaniu nie powtarza się. Ponieważ tyle dokładnie znaków tworzy początek każdego ciągu nieklasycznego z S_{G2}, nie zabraknie nigdy jego elementów do utworzenia prawidłowego rozwiązania. Liczba potrzebnych elementów zbioru S_{G2} jest określona poniższym wzorem, gdzie $n = l(|V|+2)$, $l_x = 2l-1$, l to długość elementów z S_{G1}:

$$n - l_x + 1 = l(|V|+2) - (2l-1) + 1 = l|V| + 2l - 2l + 1 + 1 = l|V| + 2 \qquad (5.19)$$

Jest to liczba elementów nieklasycznych koniecznych do uzyskania superciągu identycznego z tym, który zbuduje podzbiór elementów z S_{G1}, o ile istnieje ścieżka Hamiltona w grafie G między wierzchołkami s i t. W przykładzie tutaj rozpisanym jest to $7 \cdot 4 + 2 = 30$ elementów z S_{G2}.

Elementy nieklasyczne budujące taki superciąg nie powtarzają się, ponieważ dzięki użytej transformacji, w superciągu będącym rozwiązaniem problemu $DHPBTV$, nie powtarzają się nie tylko elementy ze zbioru S_{G1} o długości l, lecz także krótsze podciągi o długości $l-1$, od których zaczynają się elementy nieklasyczne.

Zbiór S_{G1} różni się nieco pod względem budowy elementów od zbioru w cytowanym artykule, lecz spełnione są dla niego wszystkie warunki poprawności transformacji tam przedstawione. Załóżmy, że ścieżka w grafie istnieje między wierzchołkiem s a t. Jeden element S_{G1} odpowiada każdemu wierzchołkowi, a każde $l-1$ elementów S_{G1} odpowiada każdemu łukowi w grafie. Konstrukcja elementów pozwala otrzymać ciąg $l|V|$ liter (jeżeli

wszystkie $l(|V|-1)+1$ elementów jest maksymalnie na siebie nałożonych w odpowiedniej kolejności). Po dodaniu wszystkich elementów startowych i końcowych otrzymujemy ciąg $l(|V|+1)+1$ różnych elementów spektrum. Długość superciągu to teraz $l(|V|+2)$ znaków [10].

Załóżmy teraz, że superciąg o długości $l(|V|+2)$ znaków istnieje i został zbudowany z $l(|V|+1)+1$ różnych elementów spektrum. Elementy muszą być na siebie maksymalnie nałożone, tj. na długości $l-1$. Jest to możliwe tylko wtedy, jeśli między każdą parą elementów będących odpowiednikami wierzchołków, istnieje $l-1$ elementów odpowiadających łukowi między takimi wierzchołkami w grafie. Gdyby próbować utworzyć superciąg tylko z odpowiedników wierzchołków i łuków, jego długość byłaby równa $|V|+(|V|+1)(l-1)$ ponieważ jest tylko $|V|$ wierzchołków. Taka rekonstrukcja miałaby mniej elementów niż jest to wymagane. Dlatego wprowadzone zostały elementy startowe i końcowe superciągu. Zmuszają one pierwszy odpowiednik wierzchołka w rozwiązaniu, do bycia odpowiednikiem wierzchołka s, natomiast ostatni - t. Wszystkie pozostałe elementy o długości l pomiędzy nimi w superciągu, odpowiadają innym wierzchołkom grafu pomiędzy s a t [10]. Analizowana sekwencja ma więc ściśle określoną budowę:

l elementów "startowych"

element będący odpowiednikiem s

$l-1$ elementów odpowiadających łukowi wychodzącemu z s

inne elementy reprezentujące wierzchołki i łuki w grafie

$l-1$ elementów odpowiadających łukowi wchodzącemu do t

element będący odpowiednikiem wierzchołka t

l elementów "końcowych"

Uporządkowanie elementów superciągu odpowiada ścieżce Hamiltona w problemie DHPBTV. Określono dokładną liczbę elementów ze zbioru S_{G2} potrzebną do rekonstrukcji tego samego superciągu jak wyżej. Nie mają one jednak wpływu na złożoność problemu, ponieważ superciąg opisany przed chwilą jest sam warunkiem istnienia rozwiązania problemu DHPBTV. Po tak określonej transformacji nie zabraknie jednak elementów nieklasycznych, ponieważ w procesie ich tworzenia uwzględniono wszystkie możliwe rozgałęzienia wierzchołków i łuków w grafie. Dlatego w końcowym otrzymanym spektrum instancji problemu qGSBH-ped, wszystkie niezbędne elementy nieklasyczne są obecne w S_{G2} (plus błędy pozytywne).

Prowadzi to do udowodnienia silnej NP-zupełności problemu qGSBH-ped.□

Twierdzenie 5.4.3. Problem GSBH-pes (w wersji przeszukiwania) jest silnie NP-trudny.

Dowód
Uzasadnienie powyższego twierdzenia wynika z faktu, że gdyby dla problemu przeszukiwania GSBH-pes istniał algorytm wielomianowy, to można by go wykorzystać do rozwiązania w wielomianowym czasie problemu quasiGSBH-ped [26]. Byłoby tak dlatego, że zbiór wszystkich instancji problemu przeszukiwania stanowi zbiór wszystkich instancji z odpowiedzią TAK problemu qGSBH-ped. Z tego powodu można by użyć tego algorytmu dla każdej instancji problemu quasi. Jeżeli w wielomianowym czasie algorytm nie znajdzie rozwiązania, odpowiedź dla qGSBH-ped brzmiałaby NIE, TAK w przeciwnym wypadku. W świetle teorii złożoności oraz założenia, że $P \neq NP$ nie jest to możliwe. □

5.5 Algorytm dla chipu typu Gapped

5.5.1 Dane wejściowe

Celem algorytmu jest rekonstrukcja sekwencji DNA o znanej długości n ze spektrum DNA otrzymanego z części biochemicznej metody SBH, przy użyciu nieklasycznego chipu typu Gapped. Opracowany algorytm radzi sobie z występowaniem wszystkich rodzajów błędów hybrydyzacji w spektrum, tj. błędów pozytywnych oraz błędów negatywnych. Te ostatnie mogą wynikać zarówno z pomyłek w procesie odczytu sond chipu, jak i z powodu powtórzeń podłańcuchów w badanym DNA. Dane wejściowe algorytmu są wymienione na poniższej liście:

1) spektrum DNA z eksperymentu hybrydyzacyjnego zawierające elementy pochodzące z obu połówek chipu typu Gapped;

2) długość n badanego fragmentu DNA;

3) parametr k określający między innymi długość oligonukleotydów chipu;

4) informacja o początkowym fragmencie DNA o długości k.

Parametry n oraz k pozwalają obliczyć wiele dodatkowych wartości niezbędnych dla dalszej pracy algorytmu. Parametr k pozwala określić długość oligonukleotydów w sondach klasycznych i nieklasycznych, tj. odpowiednio $l_1 = k$ oraz $l_2 = 2k - 1$. Mając daną długość DNA n, możliwe jest określenie hipotetycznej liczby elementów obu typów, którą zawierałoby spektrum idealne. Jeśli przez S^{is}_{G1} określimy zbiór elementów klasycznych

w spektrum idealnym, a przez S^{is}_{G2} zbiór elementów nieklasycznych spektrum idealnego, ich liczba w takim wypadku byłaby równa $|S^{is}_{G1}| = n - k + 1$ oraz $|S^{is}_{G2}| = n - 2k + 2$.

Gdyby możliwe było wykluczenie któregokolwiek rodzaju błędów hybrydyzacji, wtedy możliwe byłoby określenie dokładnej liczby błędów hybrydyzacji w spektrum danym. W innym wypadku algorytm musi określać tę liczbę w przybliżeniu. Jest to wymagane aby ograniczyć przestrzeń rozwiązań, którą algorytm przeszukuje, jak zostanie wyjaśnione w dalszej części tego rozdziału.

Spektrum DNA jest podzielone na dwa zbiory: zbiór S_{G1} zawierający elementy klasyczne (podciągi budujące badane DNA) oraz zbiór S_{G2}, zawierający dłuższe elementy zbudowane nad 5-literowym alfabetem. Zbiór S_{G1} jest używany do budowy grafu skierowanego. Zasada działania tej procedury jest następująca:

1) wszystkie elementy zbioru S_{G1} o długości k, opisane nad alfabetem 4-literowym, są traktowane jako wierzchołki grafu;

2) łuki w grafie tworzone są na bazie nakładania na siebie postfixu i prefixu dwóch ciągów - odpowiedników wierzchołków na długości od 1 do $k - 1$.

Postfixem o długości l_{post} ciągu e nazywamy podciąg jego ostatnich l_{post} liter. Na przykład postfixem o długości 5 ciągu opisanego jako $A\ C\ T\ C\ G\ T\ A$ jest podciąg $T\ C\ G\ T\ A$.

Prefixem o długości l_{pre} ciągu e nazywamy podciąg jego początkowych l_{pre} liter. Na przykład prefixem o długości 5 ciągu opisanego jako $G\ C\ A\ A\ C\ T\ G$ jest podciąg $GCAAC$.

Dwa ciągi e_1 i e_2 nakładają się na siebie na długości r, jeżeli postfix wierzchołka e_1 o długości r jest identyczny z prefixem wierzchołka e_2 o tej samej długości.

Dalej w pracy wyrażenia "ciąg" i "wierzchołek" będą używane zamiennie, gdyż każdy wierzchołek grafu opisany jest pewnym ciągiem znaków nad alfabetem {A,C,G,T}. Dla każdego ciągu e ze zbioru $e \in S_{G1}$ algorytm testuje występowanie wspólnych postfixów / prefixów z każdym innym ciągiem z S_{G1}. W efekcie powstaje graf skierowany, w którym połączone są ze sobą wierzchołki mające choć minimalne, jednoznakowe nałożenie. Z każdym łukiem związana jest pewna wartość "wagi", określająca liczbę znaków, którymi *nie* nakładają się na siebie dwa wierzchołki. Algorytm przyjmuje, że waga połączenia jest równa 1, jeżeli dwa wierzchołki nałożyły się na $l_1 - 1$ liter, 2 jeżeli było to $l_1 - 2$ liter, itd. W procesie rekonstrukcji DNA, czyli poszukiwania ścieżki w grafie, preferowane są łuki z jak najniższymi wagami.

Przyjmowane jest założenie, że błędy negatywne polegające na zagubieniu pewnych

elementów potrzebnych do rekonstrukcji DNA nie wystąpią jeden obok drugiego tworząc "wyrwę" o długości wiekszej niż $l_1 - 1$. Taka sytuacja uniemożliwi wtedy rekonstrukcję badanego DNA, lecz jest ona mało prawdopodobna. Druga część spektrum to zbiór S_{G2}, zawierający elementy zbudowane nad alfabetem 5-literowym. Jest on traktowany jako pewna lista, która posłuży do weryfikacji kolejno wybieranych do ścieżki wierzchołków w grafie (pochodzących ze zbioru S_{G1}).

Ostatnią ważną daną wejściową jest znajomość pierwszych k liter badanego łańcucha DNA. Pozwala ona na wiedzę o pierwszym wierzchołku e_s grafu, od którego algorytm rozpocznie poszukiwanie ścieżki, będącej odpowiednikiem rekonstrukcji sekwencji DNA.

5.5.2 Zasada działania algorytmu

W ogólności opracowany algorytm jest algorytmem dokładnym, który z uwagi na ograniczenia jak np. czas obliczeń zazwyczaj nie jest w stanie przeszukać całej przestrzeni rozwiązań. Algorytm poza najważniejszymi danymi wejściowymi biorącymi początek bezpośrednio z eksperymentu hybrydyzacyjnego, posiada także szereg parametrów regulujących jego pracę. Możliwe jest takie ich ustawienie, aby przeszukiwał on całą dostępną przestrzeń rozwiązań, z uwagi jednak na jej ogromny rozmiar, algorytm ogranicza przeszukiwanie dla pewnego dostępnego czasu obliczeń, liczby wewnętrznych iteracji czy znalezionych rozwiązań niejednoznacznych w wyznaczonym czasie. Zostanie to szczegółowe opisano w dalszej części tego podrozdziału.

W ogólności algorytm poszukuje ścieżki w grafie, pozwalającej zrekonstruować DNA o znanej długości n. Zaczynając od pierwszego wierzchołka e_s, algorytm sprawdza listę łuków prowadzących do jego następników. Lista ta jest dalej określana jako lista kandydatów *Candidates* i jest zapamiętywana dla każdego do tej pory odwiedzonego wierzchołka. Waga wybranego łuku określa też liczbę znaków, które algorytm dodaje do rekonstruowanego łańcucha DNA. Wybranie łuku o wadze większej niż 1 jest traktowane jako "zabudowanie" pewnej liczby hipotetycznych błędów negatywnych, o liczbie równej wadze pomniejszonej o 1. Po zrekonstruowaniu sekwencji DNA o ustalonej długości n, rozwiązanie takie jest dodawane do zbioru *Solutions*, algorytm zaś wycofuje się do poprzednich wierzchołków aby sprawdzić możliwość rekonstrukcji innej ścieżki, z inną możliwą kombinacją wierzchołków.

Ogólny schemat działania głównej pętli algorytmu w formie pseudokodu przedstawiony jest w formie Algorytmu 1.

Powyższy Algorytm 1 pokazuje główną pętlę, której jedno wykonanie ma na celu dodanie jednego nowego wierzchołka do budowanej ścieżki. Poniżej zostanie on opisany, natomiast ważniejsze procedury widoczne w powyższym pseudokodzie jako pojedyncze

Algorithm 1 Pseudokod głównej pętli algorytmu dla chipu typu Gapped.

```
1: while (Time < Limit AND s_space ≠ empty AND Stop = FALSE) do
2:     Candidates ← addOutgoingVerticesToList(Current_Vertex)
3:     Success_Flag ← addNewVertexToSolution(Candidates);
4:     if Success_Flag = FALSE then
5:         if Path_Length = MaxValue then
6:             Ok = testSolution(Solution);
7:             if Ok = TRUE then
8:                 addToSolutionList(Solution)
9:                 reverseSteps(Solution)
10:            else
11:                reverseSteps(Solution)
12:            end if
13:        else
14:            reverseSteps(Solution)
15:        end if
16:    end if
17: end while
```

linie poleceń zostaną dalej przedstawione w sposób dokładniejszy, z dodatkowymi sekcjami pseudokodu obrazującymi ich działanie w szczegółach.

Jeden krok algorytmu to proces dodania do aktualnie budowanej ścieżki (wektor wierzchołków *Solution*) kolejnego wierzchołka. Algorytm stara się dodawać kolejne następniki tak długo, jak jest to możliwe. W pierwszej linii zaimplementowano podstawowe warunki sprawdzane po dodaniu każdego wierzchołka do ścieżki. Algorytm sprawdza czas, jaki upłynął od momentu rozpoczęcia pracy i porównuje go z nałożonym limitem. Praca przerywana jest także po wyczerpaniu się przestrzeni rozwiązań, który to warunek ma znaczenie tylko przy stosunkowo niewielkich instancjach problemu. Zmienna *Stop* w pseudokodzie odpowiada za kilka innych warunków, które mogą być nałożone na algorytm i są sprawdzane / uaktualniane także w innych jego sekcjach, jak na przykład limit znalezionych rozwiązań czy limit nieprawidłowych rekonstrukcji mających długość badanego DNA, lecz niespełniających pewnych warunków weryfikacji. Lista wszystkich warunków zatrzymania głównej pętli to:

1) sprawdzono całą przestrzeń rozwiązań;

2) osiągnięty limit czasu przeszukiwania;

3) osiągnięty limit rozwiązań dodanych do listy rozwiązań (tzw. limit rozwiązań niejednoznacznych);

4) osiągnięty limit iteracji polegający na zbudowaniu rekonstrukcji DNA o zadanej długości, które następnie zostały odrzucone przez procedurę weryfikacji rozwiązania.

Procedura budowania listy kandydatów *Candidates*, czyli następników aktualnie odwiedzone wierzchołka, jest zapisana w linii 2. W tej fazie algorytm tworzy listę możliwych kandydatów na następników aktualnie przetwarzanego wierzchołka, czyli tego, który jako ostatni został dodany do budowanej ścieżki w grafie. W tej części następuje także weryfikacja potencjalnych nowych wierzchołków, poprzez listę elementów z nieklasycznej części spektrum, tj. ze zbioru S_{G2}.

Po przygotowaniu listy potencjalnych następników, kolejna procedura algorytmu odpowiada za próbę dodania go ścieżki (linia 3). Procedura ta jest naturalnym rozszerzeniem polecenia w linii 2, czyli przygotowania listy następników, dla zachowania jednak pewnej prostoty kodu została wydzielona jako osobna sekcja. W tej części algorytm sprawdza wiele warunków, które wierzchołek z listy następników musi spełnić, aby mógł być dodany do budowanej ścieżki. Zostaną one przedstawione dalej, w opisie procedury rozszerzania ścieżki.

W następnych liniach przedstawiono sprawdzanie zestawu warunków, mających na celu określenie dalszego toku pracy. Jeżeli wierzchołek został dodany poprawnie, iteracja pętli głównej kończy się, po czym o ile stan warunków pętli *while* nie zakończy poszukiwań, algorytm buduje nową listę kandydatów na następników, tym razem od wierzchołka właśnie dodanego. Jeśli nowy wierzchołek nie został dodany w linii 3, sprawdzane są tego powody. Jeżeli algorytm wykryje, że ostatnio dodany wierzchołek kończy ścieżkę o wielkości umożliwiającej odtworzenie DNA o odpowiedniej długości, sprawdzane jest czy ścieżka spełnia odpowiednie warunki, aby zostać zaakceptowaną jako potencjalne rozwiązanie (linia 6). Jeśli tak, algorytm dodaje ją do listy rozwiązań, po czym kontynuuje główną pętlę, w kolejnych iteracjach dodając inne wierzchołki. Po dodaniu nowego rozwiązania, a także w przypadku niemożności zaakceptowania jakiegokolwiek nowego wierzchołka z listy kandydatów, algorytm w liniach 9, 11 i 14 pseudokodu przechodzi do wydzielonej procedury mającej na celu powrót do wierzchołka, z którego możliwy był krok prowadzący do innych nieodwiedzonych jeszcze następników. Jeśli taki wierzchołek nie zostanie znaleziony, oznacza to wyczerpanie się przestrzeni rozwiązań, co implikuje koniec pracy i przejście do sekcji wygenerowania wyników rekonstrukcji DNA.

Następna procedura algorytmu, która zostanie teraz opisana, odpowiada za budowanie zbioru kandydatów na następników aktualnie odwiedzonego wierzchołka. Przedstawia ją Algorytm 2:

W linii 1 zawarta jest główna pętla całej procedury, która sprawdza wszystkie istniejące połączenia wychodzące z aktualnie dodanego do ścieżki wierzchołka. Sprawdzane

Algorithm 2 Pseudokod procedury tworzenia listy kandydatów na następników.

```
1: for (Vertex IN AllOutgoingVerticesSet) do
2:     if (checkIfUnused(vertex) = TRUE) then
3:         if (SolutionPath > MinLength AND Verification = TRUE) then
4:             Verified = verifyVertex(vertex, S_G2)
5:             if (Verified = TRUE) then
6:                 addVertexAsCandidate(vertex, Candidates)
7:             end if
8:         else
9:             addVertexAsCandidate(vertex, Candidates)
10:        end if
11:    end if
12: end for
```

są one w kolejności od największego do najmniejszego nałożenia, tj. najpierw algorytm sprawdza następniki o maksymalnym nałożeniu równym $l_1 - 1$ o wadze $w_{overlap} = 1$, następnie te nakładające się na $l_1 - 2$ znakach i wadze $w_{overlap} = 2$, itd. Taka kolejność sprawdzania i dodawania do listy kandydatów wspiera założenie preferowania przez algorytm jak największych nałożeń elementów ze spektrum.

Z każdym wierzchołkiem związana jest pewna wartość, która określa czy został on już wykorzystany w konstrukcji ścieżki. Oczywiście celem jest zbudowanie ścieżki, w której wierzchołki się nie powtarzają, dlatego tylko te jeszcze nie odwiedzone mogą zostać dodane do listy kandydatów, co sprawdza warunek w linii 2.

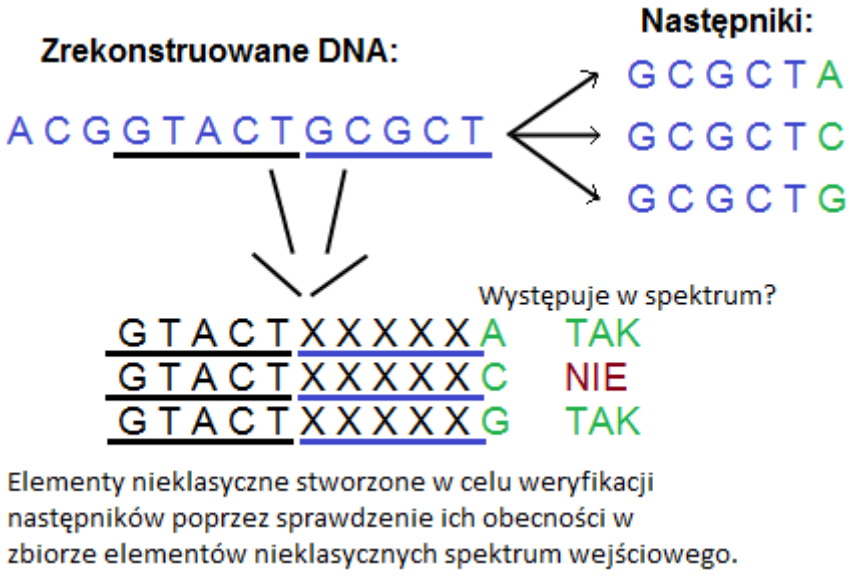
Teraz następuje sekcja kodu w liniach $3-5$, w której przeprowadzana jest weryfikacja potencjalnych następników, poprzez użycie elementów nieklasycznych spektrum - zbioru S_{G2}. Pierwszym niezbędnym warunkiem weryfikacji jest osiągnięcie minimalnej długości rekonstruowanego DNA. Sekwencja ta jest na bieżąco budowana, wraz z wydłużaniem ścieżki w grafie. Jak wyjaśniono w części rozdziału poświęconej budowie chipu typu Gapped, długość elementów nieklasycznych to $l_2 = 2k - 1$ znaków. Taka lub większa musi być suma liter do tej chwili zrekonstruowanego DNA oraz liter, które zostałyby dodane, jeśli aktualnie analizowany następnik zostałby wybrany do ścieżki.

Dla przykładu, jeśli przyjąć, że $l_2 = 2k - 1 = 13$ (dla $k = 7$), oraz aktualnie odtworzony łańcuch DNA ma $l_{DNA} = 12$ znaków, każdy wierzchołek od tego momentu może być weryfikowany. Jeśli, trzymając się przykładu, odtworzone DNA ma długość $l_{DNA} = 11$ znaków, weryfikacji poddane mogą być tylko te następniki, które mają wagę minimum $w_{overlap} = 2$, czyli w razie ich użycia dodadzą przynajmniej 2 nowe litery do rekonstruowanej sekwencji. W tym wypadku jednak zweryfikowany może być tylko ostatni nukleotyd w takim nałożeniu. Będzie o tym mowa dalej.

Jeszcze jeden warunek jest sprawdzany w linii 3, mianowicie, czy sam mechanizm

weryfikacji jest aktywny. Należy o tym wspomnieć z tego powodu, że w razie jego wyłączenia algorytm staje się czysto klasyczny, tj. operuje na danych tylko z klasycznej części spektrum. Algorytmy prezentowane w tym i dwóch następnych rozdziałach będą między innymi porównywane z podejściem klasycznym opartym na opisywanym tutaj algorytmie, jednak z całkowicie wyłączoną weryfikacją.

W linii 4 pseudokodu przedstawiono krótki zapis ustawienia flagi $Verified$ w kodzie, określającej, czy wierzchołek $vertex$ został poprawnie zweryfikowany przez użycie informacji ze zbioru elementów nieklasycznych S_{G2}. Procedura ta zostanie tutaj opisana w przykładach od najprostszego, zakładającego brak błędów hybrydyzacji, do najtrudniejszego, czyli przy ich występowaniu. Wpływ błędów pozytywnych objawiać się będzie pośrednio, co również zostanie wyjaśnione dalej. Rysunek 5.2 pomaga zrozumieć cały mechanizm.

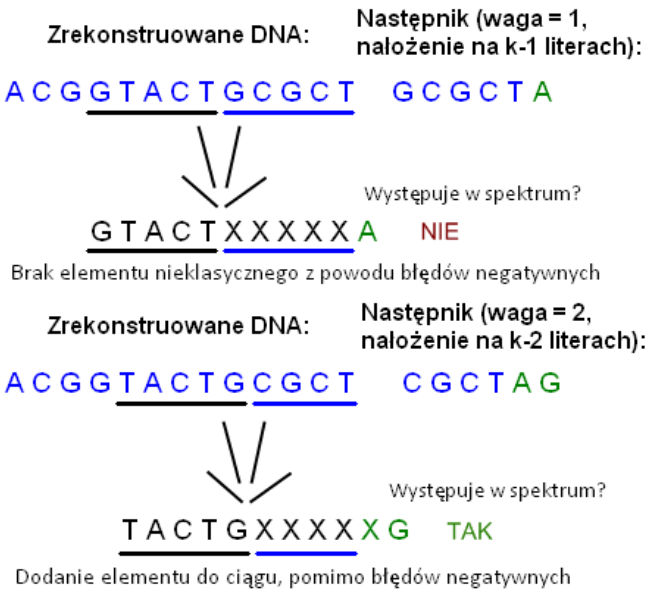


Rysunek 5.2: Weryfikacja w Gapped Chip przykład 1

W przykładzie z rysunku występuje sytuacja, w której trzy możliwe następniki nakładają się maksymalnie, na długości $l_1 - 1$, z ostatnio dodanym wierzchołkiem. Zrekonstruowane do tej pory DNA jest dłuższe niż elementy nieklasyczne, weryfikacja jest więc możliwa. Jak pokazuje rysunek, przy użyciu liter do tej pory zrekonstruowanego łańcucha DNA oraz ostatniego nukleotydu każdego następnika tworzone są hipotetyczne elementy pasujące pod względem budowy do zbioru elementów nieklasycznych spektrum. Każdy z nich jest związany z jednym rozpatrywanym następnikiem, po czym następuje sprawdzenie, które z tych elementów rzeczywiście znajdują się w zbiorze elementów nieklasycznych S_{G2}. Tylko te następniki, których nieklasyczne "odpowiedniki" przedstawione na przykładzie rzeczywiście znajdują się w tym podzbiorze spektrum,

mogą być brane pod uwagę. Dodawane są one wtedy do listy potencjalnych kandydatów. Wierzchołek $G\ C\ G\ C\ T\ C$ nie może być następnikiem, ponieważ jego dodanie do ścieżki spowodowałoby od tej chwili rekonstrukcję nieprawidłowego DNA - z uwagi na brak elementu nieklasycznego w spektrum, który mógłby zostać w tym miejscu nałożony na rekonstruowany łańcuch.

Sytuacja komplikuje się, jeśli w spektrum występują błędy negatywne. Jest to problematyczne zwłaszcza wtedy, gdy brak jest pewnych elementów nieklasycznych służących do weryfikacji. W sytuacji takiej prawidłowe (z punktu widzenia badanej sekwencji DNA) następniki w grafie mogą zostać odrzucone. Aby temu zaradzić przyjęto zasadę, w myśl której algorytm weryfikuje tylko ostatni nukleotyd następnika. W takiej sytuacji możliwe jest co prawda odrzucenie wierzchołka z tego samego powodu co właśnie opisany, jednak istnieje duża szansa, że nukleotyd (lub nukleotydy) dodawany przez taki odrzucony następnik znajdzie się w rozwiązaniu z powodu zaistnienia w innym następniku. Uniemożliwić ten mechanizm mogłoby tylko nagromadzenie się błędów negatywnych w jednym fragmencie całej badanej sekwencji DNA. Ilustruje to przykład na rysunku 5.3.



Rysunek 5.3: Weryfikacja w Gapped Chip przykład 2

Jak widać z przykładu na rysunku, zasady weryfikacji pozwalają uzyskać prawidłową rekonstrukcję pomimo błędów negatywnych uniemożliwiających poprawną weryfikację pewnych wierzchołków. Należy jednak zwrócić uwagę, że każdy błąd negatywny w zbiorze S_{G2} przekłada się na odrzucenie jakiegoś elementu ze zbioru S_{G1}. Oznacza to, że końcowa, prawidłowa rekonstrukcja używa mniej elementów z S_{G1} niż wynosiłaby liczność zbioru S_{G1}. Wartość ta jest więc pomniejszona o liczbę błędów negatywnych w zbiorze S_{G2}. Tutaj pojawia się trudność, którą pośrednio powoduje występowanie w spektrum

błędów pozytywnych obok negatywnych. Poniższa lista przedstawia przypadki występowania błędów w spektrum oraz ich wpływ na weryfikację poszczególnych elementów oraz całego rozwiązania (superciągu):

1) Brak błędów w spektrum. Możliwa jest precyzyjna weryfikacja, tj. wiadomo, że wszystkie elementy z S_{G1} muszą wystąpić w rozwiązaniu oraz zbiór S_{G2} poprawnie zweryfikuje każdy prawidłowy następnik.

2) Tylko błędy negatywne wynikające z powtórzeń występują w spektrum. Ten rodzaj błędów negatywnych nie ma wpływu na działanie procesu weryfikacji, tj. zbiór S_{G2} może poprawnie zweryfikować każdy następnik z S_{G1}. Ten rodzaj błędów ma wpływ tylko na rekonstrukcję ścieżki w grafie elementami z S_{G1}, ponieważ jest ich mniej niż powinno. Zmusza to algorytm do użycia wierzchołków z nałożeniami mniejszymi niż maksymalne, równe $l_1 - 1$. Wszystkie elementy ze zbioru S_{G1} muszą zostać użyte w rekonstrukcji DNA.

3) Błędy negatywne ogólnego rodzaju. Wpływ tego rodzaju błędów został przedstawiony na przykładzie z rysunku 5.3. W tym przypadku możliwe jest mniej precyzyjne określenie, ile elementów ze zbioru S_{G1} musi zostać użytych. Liczba ta musi być równa lub większa jego liczności, pomniejszonej o liczbę błędów negatywnych w zbiorze S_{G2}. Liczba błędów negatywnych w S_{G2} jest możliwa do wyliczenia porównując jego liczność ze zbiorem idealnym. Jeśli minimalną liczbę elementów z S_{G1}, które muszą być użyte oznaczymy jako min_{G1}, to rozwiązanie może zawierać więcej elementów niż wskazuje ta wartość. Na min_{G1} składają się bowiem błędy negatywne w zbiorze S_{G2} (które mają wpływ na odrzucanie prawidłowych elementów z S_{G1} w procesie weryfikacji), oraz błędy negatywne wynikające z powtórzeń w S_{G2}. Te ostatnie takiego odrzucania nie powodują, ale są odpowiedzialne za zmniejszenie rozmiaru zbioru S_{G2}, przy czym są nie do odróżnienia od "normalnych" błędów negatywnych.

4) Błędy negatywne oraz pozytywne w spektrum. Ten przypadek jest najtrudniejszy, lecz niestety to on właśnie jest przypadkiem realistycznym. Niemożliwe jest tutaj określenie, ile elementów ze zbioru S_{G1} musi znaleźć się w ścieżce budowanej przez algorytm. Wartość ta musi być przybliżona, aby dla potrzeb efektywności obliczeń określić, kiedy przestać korzystać z elementów zbioru S_{G1}, które nakładają się na mniej niż $l_1 - 1$ znakach. Każde takie nałożenie to "naprawa" odpowiedniej liczby "dziur" w sekwencji rekonstruowanej. Ich użycie zwiększa jednak znacznie przestrzeń rozwiązań. Dlatego od pewnego momentu algorytm stwierdza, że liczba błędów negatywnych, które zostały z pomocą takich elementów "zaklejone" musiała osiągnąć pewne maksimum, po którym dalsze rozpatrywanie elementów

nakładających się na mniejszej ilości liter przestaje być potrzebne oraz prowadzi do przeszukiwania niepotrzebnych obszarów przestrzeni rozwiązań. Nawet założenie występowania 10% czy 20% błędów negatywnych w całym spektrum jest lepsze, niż zakładanie nieograniczonej ich liczby. Jak jednak już napisano, obecność błędów pozytywnych uniemożliwia precyzyjne określenie błędów negatywnych w obu zbiorach spektrum. Dlatego też błędy pozytywne w zbiorze S_{G2}, oprócz oczywistej pozytywnej - nieprawidłowej weryfikacji powodują opisane tutaj utrudnienia dla efektywności algorytmu.

Określenie liczby błędów negatywnych występujących w spektrum na zasadnicze znaczenie dla efektywności przeszukiwania przestrzeni rozwiązań. Nowy następnik jest dodawany do ścieżki z listy kandydatów. Jeżeli jego nałożenie z ostatnim wierzchołkiem ścieżki jest mniejsze niż $l_1 - 1$ liter oznacza to, że pomaga on w naprawie efektu wystąpienia hipotetycznego błędu negatywnego, który uniemożliwił nałożenie maksymalne. Branie pod uwagę wierzchołków o nałożeniu mniejszym niż $l_1 - 1$ (teoretycznie aż do nałożenia na jednym, ostatnim/pierwszym znaku dwóch wierzchołków) powiększa przestrzeń rozwiązań w omawianym problemie. Gdy możliwe jest precyzyjne określenie liczby błędów negatywnych, tj. w przypadku gdy nie ma błędów pozytywnych, algorytm wiedząc, że "naprawił" już odpowiednią liczbę błędów w spektrum, przestaje brać pod uwagę wierzchołki z nałożeniem mniejszym niż $k - 1$ znaków, co bardzo poprawia jego efektywność. Niestety, najbardziej realistyczny przypadek pracy to ten, w którym występują wszystkie rodzaje błędów. Jak opisano powyżej, niemożliwe jest wtedy precyzyjne określenie liczby błędów negatywnych. Algorytm dla zachowania efektywności przyjmuje wartość przybliżoną, zakładając na przykład istnienie maksymalnie 20% błędów negatywnych w spektrum. Oznacza to między innymi akceptację rozwiązań, które wykorzystały przynajmniej 80% liczby elementów spektrum idealnego w danym przykładzie. Porównanie skuteczności między takim podejściem, a znajomością precyzyjnej ilości błędów negatywnych zostanie przedstawione w rozdziale z wynikami eksperymentów obliczeniowych algorytmu.

Wracając do pseudokodu procedury algorytmu, w której do listy kandydatów na następników dodawany jest kolejny wierzchołek, w liniach 5 oraz 6 jest on dodawany do listy wtedy, gdy weryfikacja zakończyła się pomyślnie. Linia 9 odpowiada za dodanie wierzchołka do listy kandydatów jeżeli weryfikacja jest wyłączona (co oznacza, że algorytm pracuje w trybie problemu sekwencjonowania klasycznego) oraz jeżeli zrekonstruowany łańcuch DNA nie osiągnął jeszcze wymaganej do weryfikacji długości. Ma to miejsce w kilku pierwszych krokach algorytmu, w których każdy następnik jest traktowany jako potencjalny prawidłowy wierzchołek ścieżki.

W sposób tutaj opisany sprawdzany jest każdy wierzchołek (poza początkowymi), co

stanowi o sile samego podejścia sekwencjonowania nieklasycznego. Ta dodatkowa informacja znacznie zmniejsza rozmiar przestrzeni rozwiązań do przeszukania.

Opisując zaproponowany algorytm, należy precyzyjniej opisać jeszcze dwie jego sekcje: powrót do poprzednich wierzchołków w celu zbadania innych rozgałęzień przestrzeni rozwiązań, a także dodanie ścieżki do listy rozwiązań. W tym ostatnim przypadku chodzi o końcową weryfikację parametrów problemu sekwencjonowania, przed akceptacją rozwiązania jako dopuszczalnego.

W liniach 9, 11 i 14 Algorytmu 1 znajduje się polecenie powrotu do poprzednich wierzchołków. Jest ono wywoływane w następujących przypadkach:

1) zostało dodane nowe rozwiązanie do listy;

2) rozwiązanie zostało odrzucone przez procedurę weryfikacji ścieżki (która zostanie opisana dalej);

3) nie powiodła się próba dodania nowego wierzchołka z listy kandydatów na następników.

Dwa pierwsze przypadki można traktować jednakowo. Niezależnie od tego czy rozwiązanie zostało zaakceptowane, ta sekcja algorytmu jest rozpatrywana tylko wtedy, gdy ścieżka osiągnęła wystarczającą długość, określoną przez rozmiar sekwencjonowanego DNA. Trzeci przypadek może być spowodowany przez wiele powodów stojących za niemożnością dodania wierzchołka do ścieżki. Lista kandydatów następników może być pusta, wierzchołki na niej zostały już użyte w ścieżce lub jedyne dostępne wierzchołki na liście kandydatów rozszerzają ścieżkę do rozmiaru, który powoduje przekroczenie rozmiaru badanego DNA. Jaki by nie był powód niemożności dodania nowego wierzchołka do konstruowanej ścieżki, opisywana teraz procedura algorytmu odpowiada za wycofanie się do wierzchołka, z którego jest możliwe rozszerzenie ścieżki niewykorzystanymi jeszcze w danej kolejności wierzchołkami.

Sama zasada działania tej części algorytmu jest dość prosta. Mając dane o aktualnie zbudowanej ścieżce oraz listy kandydatów na następników dla każdego dodanego do tej pory wierzchołka, algorytm wycofuje się do miejsca, gdzie możliwy jest wybór innego wierzchołka z list kandydatów. Jest tylko jeden wyjątek, który jest tutaj także obsługiwany: dodano wierzchołek rekonstruujący ścieżkę o zadanej długości DNA. W tym wypadku, niezależnie czy rozwiązanie zostało zaakceptowane czy nie (przypadki 1) oraz 2)), algorytm przed wycofaniem się do poprzedniego wierzchołka sprawdza, czy na ostatniej stworzonej liście kandydatów znajduje się wierzchołek, który można podmienić z aktualnie użytym tak, aby zachować wymaganą długość ścieżki (czyli musi on mieć ten sam rozmiar nałożenia co ostatnio dodany). Teoretycznie mogą wystąpić maksymalnie

4 różne następniki kończące ścieżkę, w zależności od weryfikacji oraz ich wystąpienia w części S_{G1} spektrum DNA (w przypadku spektrum z błędami hybrydyzacji). Jeśli podmiana się uda, procedura powrotu kończy pracę, a algorytm wraca do głównej pętli, w której w następnej iteracji nie doda nowego wierzchołka (ponieważ ścieżka wciąż ma maksymalny rozmiar), lecz sprawdzi, czy nowe rozwiązanie może być dodane do listy.

Jeżeli nie można zamienić ostatniego wierzchołka z innymi na aktualnej liście kandydatatów, algorytm wraca do poprzedniego odwiedzonego wierzchołka ścieżki. W tym kroku zerowane są odpowiednie wartości związane z ostatnio użytym wierzchołkiem, jak na przykład status jego użycia w ścieżce. Rekonstruowany łańcuch DNA również jest skracany o liczbę liter równą dodanej przez wycofany wierzchołek. Jeżeli wierzchołek nakładał się na mniejszej ilości znaków niż $l_1 - 1$, zmniejszany jest wewnętrzny licznik "naprawionych" skutków błędów negatywnych hybrydyzacji. Po powrocie do poprzednika usuniętego wierzchołka algorytm sprawdza, czy na liście kandydatów, na której on figurował znajduje się inny wierzchołek, który może zostać teraz użyty oraz jego dodanie jest możliwe (np. z uwagi na ilość znaków które doda do budowanego DNA). Jeśli spełnia on wszystkie warunki, jest dodawany jako następnik, procedura powrotu kończy pracę a algorytm powraca do wykonywania kolejnej iteracji pętli głównej. W innym wypadku algorytm wycofuje się dalej, aż do początków ścieżki. Jeśli procedura powrotu wykryje, że skończyła się lista kandydatów na pierwszej liście kandydatów (związanej z wierzchołkiem początkowym ścieżki), wtedy do pętli głównej algorytmu wysyłana jest informacja o przeszukaniu przestrzeni rozwiązań.

Przed dodaniem nowego rozwiązania do listy, jest ono weryfikowane. Ta procedura jest zależna od przypadku występowania lub nie dwóch głównych typów błędów hybrydyzacji. W przypadku, gdy błędy te nie występują razem, można precyzyjnie określić liczbę elementów spektrum DNA, które muszą zostać wykorzystane. Niestety, jak zostało już wcześniej podkreślone, realistyczny przypadek to ten, w którym występują wszystkie rodzaje błędów hybrydyzacji w spektrum danym na wejściu. Algorytm posługuje się wtedy przybliżoną ich wartością, domyślnie akceptując rozwiązanie, które np. zużyło minimum 80% liczby elementów spektrum idealnego, a precyzyjniej, 80% liczby elementów zbioru elementów klasycznych w spektrum idealnym.

Rozdział 6

Alternating Chip

6.1 Wstęp

Treścią niniejszego rozdziału będą problemy związane z sekwencjonowaniem przy użyciu chipu typu Alternating oraz zaproponowany algorytm, rozwiązujący problem sekwencjonowania na bazie spektrum z błędami hybrydyzacji. Drugi przedstawiony w pracy [40] chip do nieklasycznego sekwencjonowania przez hybrydyzację nosi nazwę Alternating Chip. On również używa nieokreślonych nukleotydów (tj. nukleotydów uniwersalnych) oznaczanych znakiem X we wzorze opisującym sondy. Podstawowa różnica pomiędzy nim a chipem typu Gapped zawiera się w sposobie budowy sond. O ile chip typu Gapped można było traktować jako pewną hybrydę pomiędzy podejściem klasycznym (zbiór S_{G1} spektrum GSBH) a nieklasycznym (zbiór S_{G2}), Alternating Chip składa się w całości z nieklasycznych sond. Ich budowa również dzieli chip na dwie części, jednakże w każdej połowie chipu każda sonda jest opisana wzorem nad alfabetem 5-literowym {A, C, G, T, X}, który to wzór określa zbiór oligonukleotydów w niej zawartych.

6.2 Konstrukcja i charakterystyka chipu Alternating

W alfabecie opisującym sondy chipu Alternating występuje litera X mająca takie samo znaczenie jak w przypadku chipu typu Gapped. W procesie nakładania się dwóch ciągów znaków reprezentuje ona hipotetyczny nukleotyd, do którego komplementarny jest każdy nukleotyd naturalny ze zbioru {A, C, G, T}. W przeciwieństwie jednak do chipu typu Gapped, obie połowy, na które dzieli się powierzchnia chipu Alternating, posiadają sondy opisywane w sposób nieklasyczny, tj. każda sonda chipu reprezentuje pewien zbiór oligonukleotydów opisany wzorem. Zasady konstrukcji tych wzorów w dwóch połowach chipu są następujące:

$$N_1XN_2X...XN_{k-1}XN_k \text{ oraz } N_1XN_2X...XN_{k-1}N_k$$

Zbiór spektrum zawierający sondy opisane pierwszym wzorem będzie dalej w pracy określany zbiorem S_{A1}, drugi zbiór to S_{A2}. Elementy w zbiorach mają długość:

$$l_1 = 2k - 1 \text{ oraz } l_2 = 2k - 2 \tag{6.1}$$

W przypadku braku jakichkolwiek błędów hybrydyzacji, mamy do czynienia ze spektrum idealnym. Dla długości DNA równej n, oraz ustalonego parametru k dla chipu, moce obu zbiorów to:

$$|S^{is}_{A1}| = n - (2k - 1) + 1 = n - 2k + 2 \tag{6.2}$$

$$|S^{is}_{A2}| = n - (2k - 2) + 1 = n - 2k + 3 \tag{6.3}$$

Ponieważ obie części chipu mają tę samą liczbę naturalnych nukleotydów, pojemność chipu typu Alternating jest tak samo zależna od parametru k, jak w przypadku chipu typu Gapped:

$$|C_{alt}(k)| = 2 \cdot 4^k \tag{6.4}$$

Dla przykładu rozpisane zostaną teraz przykładowe wzory sond chipu z parametrem $k = 3$, gdzie pierwsza sonda jest opisana wzorem $AXAXA$, druga natomiast to $AXAA$. Sonda pierwsza zawierać w sobie może 16 różnych oligonukleotydów, precyzyjnie:

$$\begin{array}{llll} AAAAA & ; AAACA & ; AAAGA & ; AAATA \\ ACAAA & ; ACACA & ; ACAGA & ; ACATA \\ AGAAA & ; AGACA & ; AGAGA & ; AGATA \\ ATAAA & ; ATACA & ; ATAGA & ; ATATA \end{array}$$

druga z sond, opisana przez $AXAA$ zawierać może cztery różne oligonukleotydy:

$$AAAA \ ; \ ACAA \ ; \ AGAA \ ; \ ATAA$$

Pomimo tego, że cały chip składa się z nieklasycznie skonstruowanych sond, elementy ze zbioru S_{A2} służą do weryfikacji następników wierzchołków grafu stworzonych na podstawie zbioru S_{A1} spektrum, czyli podobnie jak w przypadku algorytmu dla chipu typu Gapped. Procedura tworzenia grafu oraz samej weryfikacji zostanie opisana w dalszej części rozdziału, dotyczącej algorytmu zaproponowanego dla chipu Alternating.

6.3 Zagadnienia prawdopodobieństwa rozgałęzień

Tak jak w sekcji dotyczącej prawdopodobieństwa związanego z chipem typu Gapped, tutaj końcowe wzory opisujące charakterystykę chipu typu Alternating są identyczne, jak w odpowiednim podrozdziale rozdziału piątego. Dwa końcowe wzory opisujące prawdopodobieństwo podania rozwiązania jednoznacznego oraz maksymalną długość łańcuchów DNA, dla których chip daje odpowiedź jednoznaczną to:

$$p(C_{alt}(k), n) \approx 1 - (1 - \frac{n^2}{4^k \cdot 4^k}) \approx 3 \cdot \frac{n}{4^k} \cdot \frac{n}{4^k} = \frac{12n^2}{|C_{alt}(k)|^2} \tag{6.5}$$

$$n_{max}(C_{alt}(k), p) \approx \frac{1}{\sqrt{12}} \cdot |C_{alt}(k)|\sqrt{p} \tag{6.6}$$

6.4 Problemy oraz ich złożoność obliczeniowa

6.4.1 Podstawowe definicje

Def. 6.4.1. Ciągi typu Alternating
Ciągi typu Alternating zbudowane są nad alfabetem {A, C, G, T, X} i należą do jednego z dwóch zbiorów: zbiór S_{A1} zawiera wszystkie ciągi o długości $l_1 = 2k - 1$, zbiór S_{A2} zawiera wszystkie ciągi o długości $l_2 = 2k - 2$, gdzie $k \geqslant 3$, a elementy w obu zbiorach spełniają warunki:

1) litera X znajduje się wyłącznie na parzystych pozycjach $i = 2, 4, ..., r$ gdzie $i = 1$ oznacza pierwszy znak ciągu, $r = 2k - 2$ w przypadku zbioru S_{A1} oraz $r = 2k - 4$ w przypadku zbioru S_{A2};

2) na pozostałych pozycjach w elementach obu zbiorów występują tylko litery z podalfabetu {A, C, G, T}.

Ciągi typu Alternating są elementami zbioru $S_1 \cup S_2$

Def. 6.4.2. Nakładanie ciągów typu Alternating
Ciągi p o długości l_p oraz q o długości l_q zbudowane nad alfabetem {A, C, G, T, X} nakładają się na pewnej długości $r \leqslant min(l_p, l_q)$ jeżeli dla dwóch ich podciągów p_{a+i} oraz q_{b+i} gdzie a i b to indeks pierwszego znaku podciągu w nałożeniu odpowiednio dla p i q, $i = 0, 1, ..., r - 1$, spełnione są warunki:

1) jeżeli na pozycji i w pierwszym podciągu znajduje się litera z {A, C, G, T}, na tej samej pozycji w drugim podciągu znajduje się identyczna litera z tego alfabetu, czyli $p_{a+i} = q_{b+i}$;

2) jeżeli na pozycji i w jednym podciągu znajduje się litera X, na pozycji i w drugim podciągu może znajdować się dowolna litera z {A, C, G, T, X};

3) indeks a lub b musi być równy 1 co oznacza, że jeden z podciągów p_{a+i} lub q_{b+i} utworzony jest z początkowych r liter ciągu, od którego pochodzi.

Efektem nałożenia się p i q jest superciąg t ciągów p i q, w którym każdy znak $t_{a+b+i-1}$ (tj. znak części wspólnej p i q w t) stworzony jest następująco:

1) jeżeli $p_{a+i} = q_{b+i}$, wtedy $t_{a+b+i-1} = p_{a+i} = q_{b+i}$;

2) jeżeli $p_{a+i} = X$ oraz $q_{b+i} \in \{A, C, G, T\}$, wtedy $t_{a+b+i-1} = q_{b+i}$;

3) jeżeli $q_{b+i} = X$ oraz $p_{a+i} \in \{A, C, G, T\}$, wtedy $t_{a+b+i-1} = p_{a+i}$.

Def. 6.4.3. Uogólniony superciąg typu Alternating
Dla danego zbioru S_A ciągów typu Alternating, uogólnionym superciągiem o długości n typu Alternating jest taki superciąg, który składa się ze wszystkich elementów zbioru S_A nakładających się zgodnie ze zdefiniowaną wcześniej definicją nakładania ciągów typu Alternating.

6.4.2 Alternating SBH bez błędów hybrydyzacji

Def. 6.4.4. Idealne spektrum problemu Alternating SBH (ASBH)
Idealne spektrum składa się ze wszystkich ciągów typu Alternating o długościach $l_1 = 2k - 1$ oraz $l_2 = 2k - 2$ sekwencji DNA o długości n bez powtórzeń. W takim wypadku moce zbiorów $S_{A1}^{(is)}$ oraz $S_{A2}^{(is)}$ spełniają nierówności: $|S_{A1}^{(is)}| \leqslant n - l_1 + 1$ oraz $|S_{A2}^{(is)}| \leqslant n - l_2 + 1$.
Wszystkie ciągi ze zbiorów $S_{A1}^{(is)}$ i $S_{A2}^{(is)}$ nakładają się całkowicie na podciągi sekwencji DNA o długościach odpowiednio l_1 i l_2, zgodnie z definicją nakładania dla ciągów typu Alternating.

Problem 6.4.1. Problem ASBH bez błędów, w wersji decyzyjnej (ASBH-efd, error-free, decision)
Instancja: zbiór $S_A = S_{A1} \cup S_{A2}$ taki, że $S_{A1} = S_{A1}^{(is)}$ oraz $S_{A2} = S_{A2}^{(is)}$, będący idealnym spektrum ASBH, długość n sekwencji DNA, $|S_{A1}| = n - l_1 + 1$ oraz $|S_{A2}| = n - l_2 + 1$.
Odpowiedź: TAK, jeżeli istnieje uogólniony superciąg typu Alternating o długości n zbudowany tylko nad alfabetem {A, C, G, T}, zawierający wszystkie elementy zbiorów S_{A1} oraz S_{A2} zgodnie z definicją nakładania ciągów typu Alternating.

W tym wypadku problem decyzyjny jest obliczeniowo łatwy - wiemy, że istnieje jego rozwiązanie, z uwagi na założenie, że instancja problemu ASBH-efd składa się ze spektrum idealnego, czyli zawierającego wszystkie fragmenty DNA. Oznacza to, że zbiór wszystkich instancji problemu $D_{\Pi_{ASBH-efd}}$ jest równy zbiorowi instancji problemu, dla których odpowiedź brzmi TAK, $D_{\Pi_{ASBH-efd}} = Y_{\Pi_{ASBH-efd}}$.

Problem 6.4.2. Problem ASBH bez błędów, w wersji przeszukiwania (ASBH-efs, error-free, search)
Instancja: zbiór $S_A = S_{A1} \cup S_{A2}$ taki, że $S_{A1} = S_{A1}^{(is)}$ oraz $S_{A2} = S_{A2}^{(is)}$, będący idealnym spektrum ASBH, długość n sekwencji DNA, $|S_{A1}| = n - l_1 + 1$ oraz $|S_{A2}| = n - l_2 + 1$.
Odpowiedź: uogólniony superciąg typu Alternating o długości n zbudowany tylko nad alfabetem {A, C, G, T}, zawierający wszystkie elementy zbiorów S_{A1} oraz S_{A2} zgodnie z definicją nakładania ciągów typu Alternating.

6.4.3 Alternating SBH z błędami negatywnymi

Def. 6.4.5. Idealne multispektrum problemu Alternating SBH
Idealne multispektrum składa się ze wszystkich ciągów typu Alternating o długościach $l_1 = 2k - 1$ oraz $l_2 = 2k - 2$ z sekwencji DNA o długości n. W przypadku idealnego multispektrum moce multizbiorów $S_{A1}^{(im)}$ oraz $S_{A2}^{(im)}$ są równe odpowiednio: $|S_{A1}^{(im)}| = n - l_1 + 1$ oraz $|S_{A2}^{(im)}| = n - l_2 + 1$. Każdy element w multispektrum powtarza się tyle razy, ile odpowiadający mu podciąg w DNA.
Wszystkie ciągi ze zbiorów $S_{A1}^{(im)}$ oraz $S_{A2}^{(im)}$ zbudowane nad alfabetem 5-literowym, nakładają się całkowicie na odpowiednie podciągi o długości l_1 i l_2 z badanej sekwencji DNA, zgodnie z definicją nakładania się ciągów typu Alternating.

Problem 6.4.3. Problem ASBH z błędami negatywnymi w wersji decyzyjnej (ASBH-ned, negative error, decision)
Instancja: zbiór $S_A = S_{A1} \cup S_{A2}$ taki, że $S_{A1} \subset S_{A1}^{(im)}$ oraz $S_{A2} \subset S_{A2}^{(im)}$ będący podzbiorem bez powtórzeń idealnego multispektrum ASBH, długość n sekwencji DNA.
Odpowiedź: TAK, jeżeli istnieje uogólniony superciąg o długości mniejszej lub równej n składający się tylko liter z alfabetu {A, C, G, T}, zawierający wszystkie elementy zbiorów S_{A1} oraz S_{A2} zgodnie z definicją nakładania ciągów typu Alternating.

Zbiór wszystkich instancji powyższego problemu *nie* jest równy zbiorowi instancji, dla których odpowiedź brzmi TAK. Spektrum problemu jest podzbiorem idealnego multispektrum pochodzącego z podzielenia sekwencji DNA na mniejsze podciągi, jednakże charakter poszukiwanego rozwiązania sprawia, że istnieją takie instancje, które nie mają rozwiązania, nawet jeśli są podzbiorem multispektrum idealnego. (Na przykład: sekwencja C G A T G G A G, spektrum: CxAxG, TxGxG, CxAT - nie istnieje takie rozwiązanie,

dla którego wszystkie znaki X miałyby odpowiedniki w alfabecie {A, C, G, T}. Zbiór wszystkich instancji problemu ASBH-ned to $D_{\Pi_{ASBH-ned}} = Y_{\Pi_{ASBH-ned}} \cup N_{\Pi_{ASBH-ned}}$, gdzie $Y_{\Pi_{ASBH-ned}}$ to zbiór wszystkich instancji, dla których odpowiedź brzmi TAK, $N_{\Pi_{ASBH-ned}}$ to zbiór wszystkich instancji, dla których odpowiedź brzmi NIE.

Problem 6.4.4. Problem Alternating SBH z błędami negatywnymi w wersji przeszukiwania (ASBH-nes, negative error, search)
Instancja: zbiór $S_A = S_{A1} \cup S_{A2}$ taki, że $S_{A1} \subset S_{A1}^{(im)}$ oraz $S_{A2} \subset S_{A2}^{(im)}$ będący podzbiorem bez powtórzeń idealnego multispektrum ASBH, długość n sekwencji DNA.
Odpowiedź: uogólniony superciąg typu Alternating o długości mniejszej lub równej n składający się tylko liter z alfabetu {A, C, G, T}, zawierający wszystkie elementy zbiorów S_{A1} oraz S_{A2} zgodnie z definicją nakładania ciągów typu Alternating.

Zbiór wszystkich instancji powyższego problemu jest równy zbiorowi wszystkich instancji problemu decyzyjnego $D_{\Pi_{ASBH-nes}} = D_{\Pi_{ASBH-ned}}$

6.4.4 Alternating SBH z błędami pozytywnymi

Definicje problemów sformułowane poniżej wykorzystują definicję idealnego spektrum ASBH 6.4.4 oraz idealnego multispektrum dla ASBH 6.4.5.

Problem 6.4.5. Problem Alternating SBH z błędami pozytywnymi, w wersji decyzyjnej (ASBH-ped, positive error, decision)
Instancja: zbiór $S_A = S_{A1} \cup S_{A2}$ taki, że $S_{A1}^{(is)} = S_{A1}^{(im)} \subset S_{A1}$ oraz $S_{A2}^{(is)} = S_{A2}^{(im)} \subset S_{A2}$ zawierające w sobie idealne spektrum ASBH, długość n sekwencji DNA.
Odpowiedź: TAK, jeżeli istnieje uogólniony superciąg zbudowany tylko nad alfabetem {A, C, G, T} o długości n zawierający $|S_{A1}^{(is)}|$ różnych elementów ze zbioru S_{A1} oraz $|S_{A2}^{(is)}|$ różnych elementów ze zbioru S_{A2}.

Zbiór wszystkich instancji powyższego problemu jest równy zbiorowi instancji, dla których odpowiedź brzmi TAK: $D_{\Pi_{ASBH-ped}} = Y_{\Pi_{ASBH-ped}}$. Idealne spektrum problemu ASBH jest podzbiorem spektrum powyższego problemu. Wiadomo, że w idealnym spektrum ASBH istnieje uogólniony superciąg o długości n zbudowany nad alfabetem {A, C, G, T}, tak więc złożoność powyższego problemu decyzyjnego jest wielomianowa: odpowiedź zawsze brzmi TAK.

Problem 6.4.6. Problem ASBH z błędami pozytywnymi w wersji przeszukiwania (ASBH-pes, positive error, search)
Instancja: zbiór $S_A = S_{A1} \cup S_{A2}$ taki, że $S_{A1}^{(is)} = S_{A1}^{(im)} \subset S_{A1}$ oraz $S_{A2}^{(is)} = S_{A2}^{(im)} \subset S_{A2}$ zawierające w sobie idealne spektrum ASBH, długość n sekwencji DNA.

Odpowiedź: uogólniony superciąg zbudowany tylko nad alfabetem {A, C, G, T} o długości n zawierający $|S_{A1}^{(is)}|$ różnych elementów ze zbioru S_{A1} oraz $|S_{A2}^{(is)}|$ różnych elementów ze zbioru S_{A2}.

Zbiór wszystkich instancji tego problemu jest równy zbiorowi instancji problemu decyzyjnego: $D_{\Pi_{ASBG-pes}} = D_{\Pi_{ASBH-ped}}$. Należy dodać, że jest on przez to równy zbiorowi instancji z odpowiedzią TAK dla problemu decyzyjnego: $D_{\Pi_{ASBG-pes}} = Y_{\Pi_{ASBH-ped}}$.

Problem 6.4.7. Problem quasi-pozytywnego ASBH w wersji decyzyjnej (qASBH-ped, positive errors, decision)
Instancja: zbiory S_{A1} i S_{A2} będące spektrum ASBH, zawierające tylko ciągi typu Alternating o długościach l_1 i l_2, długość n badanej sekwencji DNA.
Odpowiedź: TAK, jeżeli istnieje uogólniony superciąg zbudowany tylko nad alfabetem {A, C, G, T} o długości n zawierający $|S_{A1}^{(is)}|$ różnych elementów ze zbioru S_{A1} oraz $|S_{A2}^{(is)}|$ różnych elementów ze zbioru S_{A2}, gdzie $|S_{A1}^{(is)}| = n - l_1 + 1$ i $|S_{A2}^{(is)}| = n - l_2 + 1$ to liczność podzbiorów idealnego spektrum ASBH.

Zbiór wszystkich instancji z odpowiedzią TAK problemu qASBH-ped jest równy zbiorowi wszystkich instancji z odpowiedzią TAK dla problemu ASBH-ped. Zbiór wszystkich instancji ASBH-ped jest jednak podzbiorem zbioru wszystkich instancji problemu qASBH-ped, ponieważ ten drugi zawiera także instancje, dla których odpowiedź brzmi NIE, $Y_{\Pi_{qASBH-ped}} = Y_{\Pi_{ASBH-ped}}$, $D_{\Pi_{ASBH-ped}} \subset D_{\Pi_{qASBH-ped}}$

Twierdzenie 6.4.1. Problem qASBH-ped jest silnie NP-zupełny.

Dowód
Punktem wyjścia w dowodzie powyższego twierdzenia będzie wykorzystanie zmodyfikowanej transformacji dla problemu chipu typu Gapped z błędami pozytywnymi. Modyfikowana jest więc tutaj po raz kolejny transformacja z artykułu [10].
W pierwszej kolejności należy jednak sformułować problem początkowy, dla którego transformacja ta będzie stosowana.

Problem 6.4.8. Problem ścieżek Hamiltona w dwóch grafach skierowanych (DHP, Double Hamiltonian Paths)
Instancja: dane są dwa skierowane grafy $G_1 = (V_1, A_1)$ i $G_2 = (V_2, A_2)$ mające taką samą liczbę wierzchołków, w obu grafach określone są wierzchołki startowe s_1 i s_2 oraz końcowe t_1 i t_2.
Odpowiedź: TAK, jeżeli w grafie pierwszym i drugim istnieją ścieżki Hamiltona pomiędzy określonymi w nich wierzchołkami startowymi i końcowymi.

Specyficzną wersją tego problemu jest silnie NP-zupełny problem odnalezienia skierowanej ścieżki Hamiltona pomiędzy dwoma wierzchołkami (DHPBTV), stosowany już wcześniej w poprzednim rozdziale.

Problem 6.4.9. Skierowana ścieżka Hamiltona między dwoma wierzchołkami, DHPBTW
Instancja: graf skierowany $H = (V, A)$, z zaznaczonym wierzchołkiem początkowym i końcowym s i t.
Odpowiedź: TAK, jeżeli istnieje ścieżka Hamiltona pomiędzy s i t.

Jedynym przejętym założeniem jest to, aby grafy miały tę samą liczbę wierzchołków, a co za tym idzie ścieżki Hamiltona o równej długości. Poza tym jednym ograniczeniem nic nie łączy obu grafów. Oczywistym jest, że takie założenie nie ma wpływu na złożoność nowego problemu. Do zakodowania wierzchołków i ścieżek w grafach zostanie użyta podobna transformacja, co opisana dla problemu Gapped SBH z błędami pozytywnymi. Ma ona pewną cechę charakterystyczną, która umożliwi poprawne odtworzenie podzbioru S_{A2} spektrum problemu qASBH-ped. Cecha ta polega na tym, że superciąg wynikowy (utworzony jako odpowiednik prawidłowego rozwiązania problemu ścieżek Hamiltona) zbudowany jest z elementów o długości l, to nie powtarzają się w nim podciągi krótsze - o długości $l-1$. Będzie to istotne w kontekście tworzenia podzbioru S_{A2} dla spektrum qASBH-ped.

Transformacja o której mowa (zmodyfikowana transformacja z [10] dla problemu chipu Gapped z błędami pozytywnymi) zostanie tutaj użyta dwa razy, za drugim razem w zmienionej formie do zakodowania drugiego grafu. Najpierw jednak, poniżej, zaprezentowana została transformacja dla problemu chipu Gapped z błędami pozytywnymi, która zakoduje pierwszy graf instancji problemu wyjściowego.

Mając daną instancję DHPBTV (Directed Hamiltonian Path Between Two Vertices) postępujemy następująco:

1) każdemu wierzchołkowi $v \in V$ przydzielamy unikalną nazwę $e(v)$ o długości $\lceil log_2|V| \rceil$ nad alfabetem {A, C};

2) definiujemy $l = 2\lceil log_2|V| \rceil + 3$ jako długość wszystkich elementów w S;

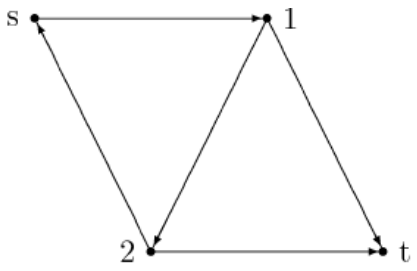
3) budujemy zbiór S tak, że dla każdego wierzchołka $v \in V$ tworzony jest jeden odpowiadający mu ciąg znaków $e(v) \cdot G \cdot e(v) \cdot TT$ ($\cdot$ oznacza konkatenację, G oraz T to nukleotydy);

4) do S dodajemy $l-1$ oligonukleotydów dla każdego łuku $(u, v) \in A$, oligonukleotydy te to ciągi znaków: $u_2 \cdot u_3 \cdot \cdots \cdot u_l v_1, u_3 \cdot u_4 \cdot \cdots \cdot v_1 \cdot v_2, \ldots, u_l \cdot v_1 \cdot \cdots \cdot v_{l-2} \cdot v_{l-1}$ (gdzie $u_1 \cdot u_2 \cdot \cdots \cdot u_l$ to oligonukleotyd oznaczający wierzchołek u);

5) do S dodajemy l oligonukleotydów startowych w formie ciągów znaków: $t_1 \cdot g_2 \cdot \cdots \cdot g_{l-1}, g_2 \cdot g_3 \cdot \cdots \cdot g_l \cdot s_1, \ldots, g_l \cdot s_1 \cdot \cdots \cdot s_{l-2} \cdot s_{l-1}$ (gdzie $t_1 = T, g_i = G, 2 \leqslant i \leqslant l$ oraz $s_1 \cdot s_2 \cdot \cdots \cdot s_l$ to oligonukleotyd reprezentujący wierzchołek startowy s);

6) do S dodajemy l oligonukleotydów końcowych w formie ciągów znaków: $t_2 \cdot t_3 \cdot \cdots \cdot t_l \cdot k_1, t_3 \cdot t_4 \cdot \cdots \cdot k_1 \cdot k_2, \ldots, t_l \cdot k_1 \cdot \cdots \cdot k_{l-2} \cdot k_{l-1}, k_1 \cdot k_2 \cdot \cdots \cdot k_l, k_2 \cdot k_3 \cdot \cdots \cdot k_{l-1} \cdot G, k_3 \cdot k_4 \cdot \cdots \cdot k_{l-2} \cdot G \cdot T$ (gdzie $k_i = T, 1 \leqslant i \leqslant l-1$ oraz $t_1 \cdot t_2 \cdot \cdots \cdot t_l$ to oligonukleotyd reprezentujący wierzchołek końcowy t).

Powyższa transformacja umożliwia utworzenie instancji problemu klasycznego SBH z błędami pozytywnymi. Zbiór będący spektrum instancji takiego problemu powstaje poprzez umieszczenie w nim tylko niepowtarzających się łańcuchów znaków powstałych po transformacji. Powtórzenia mogą wystąpić tylko wtedy, jeśli z danego wierzchołka wychodzą łuki do dwóch lub więcej wierzchołków lub więcej niż jeden łuk wchodzi do jednego wierzchołka. Aby udowodnić prawidłowość transformacji zostało w [10] wykazane, że wspomniana wcześniej ścieżka Hamiltona istnieje wtedy i tylko wtedy, gdy istnieje sekwencja (superciąg) o długości $n = l(|V| + 2)$ utworzona z elementów spektrum, zawierająca $l(|V|+1)+1$ jego różnych elementów. Jak wyjaśniono już w rozdziale poprzednim, jedyna do tego momentu modyfikacja, polegająca na dodaniu dodatkowego znaku pod koniec każdego ciągu kodującego wierzchołki grafu, nie ma wpływu na poprawność przytoczonego podejścia. Przyjmijmy przykładowy graf z pracy [10] dany na rysunku 6.1:



Rysunek 6.1: Graf z instancji problemu DHPBTW [10]

Dla takiego grafu powstaje następujący zbiór S_1 elementów:

1) Odpowiedniki wierzchołków (kolejno s, 1, 2, t):
$AAGAATT$, $ACGACTT$, $CAGCATT$, $CCGCCTT$

2) Elementy prowadzące do wierzchołka s:
$TGGGGGG$, $GGGGGGA$, $GGGGGAA$, $GGGGAAG$, $GGGAAGA$, $GGAAGAA$, $GAAGAAT$ (podzbiór startowy)

3) Elementy wychodzące z wierzchołka t:
$CGCCTTT$, $GCCTTTT$, $CCTTTTT$, $CTTTTTT$, $TTTTTTT$, $TTTTTTG$, $TTTTTGT$ (podzbiór końcowy)

4) Elementy tworzące łuki w grafie (ścieżki łukowe):
(s,1) - $AGAATTA$, $GAATTAC$, $AATTACG$, $ATTACGA$, $TTACGAC$, $TACGACT$
(1,2) - $CGACTTC$, $GACTTCA$, $ACTTCAG$, $CTTCAGC$, $TTCAGCA$, $TCAGCAT$
(1,t) - $CGACTTC$, $GACTTCC$, $ACTTCCG$, $CTTCCGC$, $TTCCGCC$, $TCCGCCT$
(2,s) - $AGCATTA$, $GCATTAA$, $CATTAAG$, $ATTAAGA$, $TTAAGAA$, $TAAGAAT$
(2,t) - $AGCATTC$, $GCATTCC$, $CATTCCG$, $ATTCCGC$, $TTCCGCC$, $TCCGCCT$

W instancji problemu 6.4.8 mowa jest o dwóch grafach skierowanych. Drugi graf musi mieć tę samą liczbę wierzchołków co pierwszy i jest to jedyny warunek "łączący" grafy. W przykładzie rozwijanym w tej sekcji przyjmijmy dwa identyczne grafy, tj. z tą samą ścieżką Hamiltona. Paradoksalnie nie zmieni to niczego, ponieważ już jednokrotne znalezienie ścieżki Hamiltona w dowolnym omawianym tutaj grafie jest trudne, nie ma więc dla potrzeb przykładu różnicy czy graf drugi jest identyczny czy nie. Dzięki takiemu podejściu łatwiej będzie pokazać, dlaczego przy wspomnianej już modyfikacji powyższej transformacji dowolny drugi graf (nawet identyczny z pierwszym) zostaje zakodowany w formie ciągów liter w sposób całkowicie różny od pierwszego. Innymi słowy zbiory ciągów znaków dla obu grafów pozostają rozłączne, pomimo zapisu na tym samym alfabecie. Zmiana sposobu kodowania polega na zastąpieniu w powyższej transformacji każdej litery według schematu:

1) A na G

2) C na T

3) G na A

4) T na C

Efektem jest poniższa transformacja dla drugiego grafu. Mając dany drugi z grafów problemu 6.4.8 postępujemy następująco:

1) każdemu wierzchołkowi $v \in V$ przydzielamy unikalną nazwę $e(v)$ o długości $\lceil log_2|V| \rceil$ nad alfabetem {G, T};

2) definiujemy $l = 2\lceil log_2|V|\rceil + 3$ jako długość wszystkich elementów w S;

3) budujemy zbiór S tak, że dla każdego wierzchołka $v \in V$ budujemy jeden odpowiadający mu ciąg znaków $e(v) \cdot A \cdot e(v) \cdot CC$ ($\cdot$ oznacza konkatenację, A oraz C to nukleotydy);

4) do S dodajemy $l-1$ oligonukleotydów dla każdego łuku $(u, v) \in A$, oligonukleotydy te to ciągi znaków $u_2 \cdot u_3 \cdot \cdots \cdot u_l v_1, u_3 \cdot u_4 \cdot \cdots \cdot v_1 \cdot v_2, \ldots, u_l \cdot v_1 \cdot \cdots \cdot v_{l-2} \cdot v_{l-1}$ (gdzie $u_1 \cdot u_2 \cdot \cdots \cdot u_l$ to oligonukleotyd oznaczający wierzchołek u);

5) do S dodajemy l oligonukleotydów startowych w formie ciągów znaków: $t_1 \cdot g_2 \cdot \cdots \cdot g_{l-1}, g_2 \cdot g_3 \cdot \cdots \cdot g_l \cdot s_1, \ldots, g_l \cdot s_1 \cdot \cdots \cdot s_{l-2} \cdot s_{l-1}$ (gdzie $t_1 = C, g_i = A, 2 \leqslant i \leqslant l$ oraz $s_1 \cdot s_2 \cdot \cdots \cdot s_l$ to oligonukleotyd reprezentujący wierzchołek startowy s);

6) do S dodajemy l oligonukleotydów końcowych w formie ciągów znaków $t_2 \cdot t_3 \cdot \cdots \cdot t_l \cdot k_1, t_3 \cdot t_4 \cdot \cdots \cdot k_1 \cdot k_2, \ldots, t_l \cdot k_1 \cdot \cdots \cdot k_{l-2} \cdot k_{l-1}, k_1 \cdot k_2 \cdot \cdots \cdot k_l, k_2 \cdot k_3 \cdot \cdots \cdot k_{l-1} \cdot A, k_3 \cdot k_4 \cdot \cdots \cdot k_{l-2} \cdot A \cdot C$ (gdzie $k_i = C, 1 \leqslant i \leqslant l-1$ oraz $t_1 \cdot t_2 \cdot \cdots \cdot t_l$ to oligonukleotyd reprezentujący wierzchołek końcowy t).

Przykład transformacji dla tego samego grafu rozpisany został poniżej, jako zbiór S_2:

1) odpowiedniki wierzchołków (kolejno s, 1, 2, t):
$GGAGGCC$, $GTAGTCC$, $TGATGCC$, $TTATTCC$

2) elementy prowadzące do wierzchołka s:
$CAAAAAA$, $AAAAAAG$, $AAAAAGG$, $AAAAGGA$, $AAAGGAG$, $AAGGAGG$, $AGGAGGC$ (podzbiór startowy)

3) elementy wychodzące z wierzchołka t:
$TATTCCC$, $ATTCCCC$, $TTCCCCC$, $TCCCCCC$, $CCCCCCC$, $CCCCCCA$, $CCCCCAC$ (podzbiór końcowy)

4) elementy tworzące łuki grafu (ścieżki łukowe):
(s,1) - $GAGGCCG$, $AGGCCGT$, $GGCCGTA$, $GCCGTAG$, $CCGTAGT$, $CGTAGTC$
(1,2) - $TAGTCCT$, $AGTCCTG$, $GTCCTGA$, $TCCTGAT$, $CCTGATG$, $CTGATGC$
(1,t) - $TAGTCCT$, $AGTCCTT$, $GTCCTTA$, $TCCTTAT$, $CCTTATT$, $CTTATTC$
(2,s) - $GATGCCG$, $ATGCCGG$, $TGCCGGA$, $GCCGGAG$, $CCGGAGG$, $CGGAGGC$
(2,t) - $GATGCCT$, $ATGCCTT$, $TGCCTTA$, $GCCTTAT$, $CCTTATT$, $CTTATTC$

Różnica w stosunku do zbioru S_1 będzie widoczna, gdy dla użytego przykładu grafu skierowanego zostaną rozpisane dwa superciągi stanowiące rozwiązania (sekwencja wierzchołków: -starter- -s- -1- -2- -t- -koniec-):

```
TGGGGGG AAGAATT ACGACTT CAGCATT CCGCCTT TTTTTGT (superciąg I)
CAAAAAA GGAGGCC GTAGTCC TGATGCC TTATTCC CCCCCAC (superciąg II)
```

Pomimo, że ciągi znaków wydłużą się w przypadku większych grafów, a także mimo tego, że w obu grafach ścieżki Hamiltona mogą przebiegać w dowolny sposób, zostaje wciąż zachowanych kilka zasad określających tworzenie dowolnych podciągów o długości l w obu superciągach. Przede wszystkim w przypadku pierwszego kodowania litery G oraz T wystąpią obok siebie tylko w dwóch miejscach, dla dowolnych możliwych odpowiedników ścieżek Hamiltona zawsze w tych samych: na samym początku pierwszego elementu superciągu oraz na samym końcu jego ostatniego elementu. We wszystkich innych miejsach litery te nigdy nie występują obok siebie, oraz dzieli jest zawsze ściśle określona odległość. W opisie drugiego grafu zasada ta dotyczy liter A oraz C. Jak widać, dwa ciągi zbudowane na tych przeciwstawnych zasadach, nie mają elementów wspólnych o długości l. Tak samo zachowana jest zasada wynikająca z pierwotnej modyfikacji transformacji, wprowadzonej już w poprzednim rozdziale. Mianowicie, podciągi o długości $l-1$ nie powtarzają się ani w ramach jednego superciągu, ani we wspólnym zbiorze elementów.

Zamiana zbioru $S = S_1 \cup S_2$ ciągów l literowych ($l = 7$ dla omawianego przykładu) na zbiór S_{A1} spektrum jest bardzo prosta. Każdy element zbioru S jest zamieniany na element typu Alternating o długości $l_1 = 2l - 1$ poprzez wstawienie znaków X pomiędzy każdą parę liter tworzących dany ciąg. Dla przykładu odpowiednik wierzchołka s w pierwszym zbiorze (grafie) to $AAGAATT$, po zamianie przyjmuje on formę $AXAXGXAXAXTXT$. W zbiorach S_1 i S_2 mogły wystąpić powtórzenia z dokładnie tych samych powodów, jak opisano w poprzednim rozdziale. Powodem może być więcej niż jeden łuk wchodzący lub wychodzący z wierzchołków grafu. Transformacja zapewnia jednak, że superciąg wynikowy nie będzie miał powtórzeń swoich podciągów o długości $2l-1$ (ponieważ dodanie znaków X niczego tu nie zmienia). Wystarczy więc usunąć powtarzające się elementy w zbiorze $S = S_1 \cup S_2$, aby otrzymać jeden zbiór S_{A1} spektrum problemu ASBH-pes.

Elementy o długości l_1 z S_{A1}, kodujące oba grafy, są od siebie różne pod względem budowy. Także elementy krótsze, o długości $l_1 - 2$, nie powtarzają się w superciągach. Niemożliwe jest więc nałożenie dwóch elementów pochodzących z różnych grafów (po transformacji opisanej wyżej) na $l_1 - 2$ znakach. Oczywiście dwa elementy mogą się wtedy nałożyć na znakach X (nałożenie na $l_1 - 1$ znakach) zgodnie z regułą nakładania. Jeśli jednak nastąpi to dla elementów kodujących tylko jeden graf, niemożliwe stanie się

uzyskanie superciągu o odpowiedniej długości dla tego grafu (będzie to dokładnie wyjaśnione dalej). Z drugiej strony, elementy pochodzące z dwóch różnych grafów, mogą się w rozwiązaniu prawidłowym nałożyć tylko na znakach X, jak np. $AXAXA$ z $CXCXC$ tworzące $ACACAC$. Takie dwa elementy (pochodzące z transformacji z dwóch grafów, nie z tego samego) nie mogą się nałożyć maksymalnie, tj. na $l_1 - 2$ literach (tj. quasi-maksymalnie: uwzględniając rozdzielające X), ponieważ podciągi o takiej długości nie wystąpią w elementach dwóch wersji transformacji (z zamianą liter kodujących dla drugiego grafu). Oznacza to, że pomimo wymieszania w zbiorze S_{A1} spektrum problemu ASBH elementów kodujących pierwszy i drugi graf, ich nałożenie ze sobą (tj. nałożenie quasi-maksymalne) nie będzie możliwe.

Cała powyższa część transformacji kończy się połączeniem dwóch zbiorów S_1 i S_2 w jeden, dodaniem rozdzielających znaków X pomiędzy litery z {A,C,G,T} jego elementów oraz usunięciem powtórzeń. Efektem jest wtedy zbiór S_{A1}. Powtórzenia elementów w S_1 i S_2 (niezależnie w każdym zbiorze) są efektem tylko i wyłącznie tego, że w grafach z wierzchołków może wychodzić więcej niż jeden łuk, a także więcej niż jeden łuk może kierować do jednego wierzchołka. Nie są one w stanie wystąpić na ścieżce będącej ścieżką Hamiltona (ponieważ odwiedza ona dany wierzchołek tylko raz) ani w jednym, ani w drugim grafie. Powtórzenia z powodu kodowania dwóch grafów nie są możliwe, co zostało już opisane. Innymi słowy po połączeniu zbiorów oraz usunięciu jedynych dopuszczalnych powtórzeń ciągów powstaje zbiór S_{A1} instancji problemu pozytywnego quasi-ASBH.

Aby otrzymać pełną instancję problemu qASBH-ned, należy jeszcze zbudować jej podzbiór S_{A2}. Jego elementy tworzone są w następujący sposób:

1) brane jest zawsze pierwsze $l_1 - 2$ liter każdego elementu e ze zbioru S_{A1} - powstaje podciąg *str*;

2) dla każdego podciągu *str* tworzone są cztery różne ciągi poprzez dodanie do *str* kolejno liter A, C, G oraz T;

3) wszystkie cztery ciągi utworzone w poprzednim punkcie są dodawane do zbioru S_{A2}.

Modyfikacja transformacji wprowadzona już dla chipu typu Gapped w poprzednim rozdziale, zapewnia niepowtarzalność w superciągu-rozwiązaniu elementów krótszych o dwa znaki w stosunku do pełnej długości elementów S_{A1}. W poprzedniej transformacji (problemu quasi-GSBH-ped) nie powtarzały się podciągi $l - 1$ pierwszych znaków. W przypadku omawianej instancji problemu qASBH-ped, której elementy zawierają rozdzielający znak X, nie powtarzają się elementy o długości $l_1 - 2$, gdzie l_1 to długość

ciągów zbioru S_{A1}. Dzięki temu kolejne utworzone elementy ze zbioru S_{A2} zgodnie z powyższymi zasadami również pozostają unikalne w swoim zbiorze.

Pod koniec wszystkich przekształceń otrzymujemy spektrum problemu qASBH-ned z błędami pozytywnymi. Ponieważ w S_{A1} znajdują się ciągi kodujące dwa grafy, oraz jak wyjaśniono wcześniej, elementy kodujące nie powtarzają się między sobą, możliwe byłoby uzyskanie ze zbioru S_{A1} dwóch superciągów o równej długości. Byłyby one odpowiednikami ścieżek Hamiltona w grafie, o ile te istnieją. W rzeczywistości reguła nakładania zapewnia podobny efekt, z tym, że znaki X służą do nałożenia się dwóch superciągów na siebie. Powstaje wtedy superciąg, w którym nieparzyste i parzyste znaki odczytywane kolejno kodują dwa superciągi składowe, będące odpowiednikami ścieżek Hamiltona z wyjściowego problemu.

Rozwiązaniem problemu qASBH-ped jest pewien uogólniony superciąg, zbudowany zgodnie z regułami nakładania dla chipów typu Alternating, składający się:

1) $l_1(|V|+2)+|V|+2$ znaków (tylko alfabetu {A, C, G, T}, co zapewnia reguła nakładania);

2) $(l_1+1)(|V|+1)+2)$ elementów z części S_{A1} spektrum;

3) $(l_2+2)(|V|+1)+3$ elementów z części S_{A2} spektrum.

Rozwiązaniem problemu po transformacji jest *jeden* uogólniony superciąg o długości $l_1(|V|+2)+|V|+1+1$ znaków. Aby łatwiej zrozumieć jego genezę, w przykładzie był traktowany jak dwa superciągi składowe, razem określające jego litery nieparzyste i parzyste. Każdy z tych dwóch superciągów określa ścieżkę Hamiltona dla odpowiedniego grafu. Kolejno są w nich więc położone: ciąg-starter, odpowiednik wierzchołka s, kolejne wierzchołki grafu, odpowiednik wierzchołka t oraz element końcowy. Graf ma $|V|$ wierzchołków, ich odpowiedniki z S_{A1} mają długość l_1 po transformacji. Razem z elementem startowym i końcowym daje to długość $l_1(|V|+2)$ znaków. Do tego dochodzą znaki X pomiędzy sześcioma wymienionymi (dla używanego przykładu) elementami, jest ich zawsze $|V|+1$. Ostatni składnik sumy, czyli ostatnia dodana we wzorze $l_1(|V|+2)+|V|+1+1$ liczba 1 uwzględnia długość uogólnionego superciągu, gdy jego dwa superciągi składowe nałożą się na swoich znakach X (w czysto teoretycznym przypadku budowania ich osobno).

$(l_1+1)(|V|+1)+2$ to liczba elementów z części S_{A1} spektrum, powstała z użycia wzoru $n-l+1$ na liczbę podciągów, przy założeniu, że n ma długość $l_1(|V|+2)+|V|+2$. Nie ma możliwości nałożenia się na l_1-2 znakach jakiejkolwiek pary elementów kodujących dwa grafy z instancji problemu DHP. Innymi słowy można stworzyć dwa niezależne superciągi, składające się z połowy podanej liczby elementów każdy, w których

każda litera z {A, C, G, T} jest rozdzielona literą X. Nałożenie tych dwóch superciągów na siebie na znakach X stworzy jeden uogólniony superciąg o długości o jeden większej niż długość superciągów składowych. Na przykład, dwa superciągi $AxAxAxA$ oraz $CxCxCxC$ o długości 7 znaków każdy tworzą 8-znakowy superciąg $ACACACAC$ (lub $CACACACA$) będący odpowiednikiem ścieżek Hamiltona w obu grafach, zależnie czy odczytujemy litery parzyste czy nieparzyste.

Kwestia elementów ze zbioru S_{A2} spektrum jest nieco bardziej skomplikowana. Muszą one być w stanie nałożyć się na ten sam, uogólniony superciąg opisany wyżej, w liczbie $(l_2 + 2)(|V| + 1) + 3$ elementów zbioru S_{A2}. Ponieważ liczba wszystkich podciągów o długości l pewnego superciągu o długości n nakładających się maksymalnie to $n - l + 1$, podstawiając: $n = l_1(|V|+2)+|V|+2$, $l = l_2$, oraz $l_1 = 2k-1$, $l_2 = 2k-2$ otrzymujemy:

$2k(|V| + 1) + 3$, czyli $(l_2 + 2)(|V| + 1) + 3$

Suma o której tu mowa, to liczba elementów z S_{A2}, które z nałożeniem co jeden znak należy nałożyć na uogólniony superciąg zgodnie z regułą nakładania typu Alternating.

Podsumowując całość zagadnień, możemy wymienić cechy charakterystyczne transformacji:

1) Koduje ona na tym samym alfabecie {A, C, G, T, X} dwa grafy z instancji problemu wyjściowego, na ciągi znaków zbioru S_{A1} w taki sposób, że nie powtarzają się żadne ciągi pomiędzy dwoma zbiorami kodującymi grafy. Poprzez modyfikację znaków alfabetu dla transformacji jednego i drugiego grafu, jedyne powtórzenia jakie mogą się pojawić, występują w obrębie każdego z podzbiorów. Może być to spowodowane strukturą połączeń w grafie o czym była już mowa.

2) Niezależnie od liczby ścieżek Hamiltona w obu grafach, elementy ze zbioru S_{A2} zawsze nałożą się (w odpowiedniej, określonej liczbie) na jeden uogólniony superciąg wynikowy, kodujący dwie ścieżki Hamiltona w dwóch grafach.

3) Liczba elementów z S_{A1} i z S_{A2} potrzebnych do pełnego nałożenia się ich na prawidłowy superciąg wynikowy jest ściśle określona. Długość superciągu będącego odpowiednikiem ścieżek Hamiltona jest również określona.

Widać z powyższego wywodu, że elementy ze zbioru S_{A2} nie są w zasadzie potrzebne do budowy superciągu - weryfikują go tylko poprzez pełne nałożenie się na całej jego długości zgodnie z regułami nakładania ciągów typu Alternating, w ściśle określonej liczbie. Poprawność transformacji (odwzorowanie rozwiązań ścieżek Hamiltona w grafach) leży więc w zasadach użycia elementów ze zbioru S_{A1}.

Dla tego samego grafu użytego w przykładzie, kodowanego raz jednym, raz drugim sposobem, superciągi składowe wraz ze złożeniem (w uogólniony superciąg) to:

```
TxGxGxGxGxGxGxAxAxGxAxAxTxTxAxCxGxAxCxT...CxCxGxCxCxTxTxTxTxTxTxGxT
 CxAxAxAxAxAxAxGxGxAxGxGxCxCxGxTxAxGxT...CxTxTxAxTxTxCxCxCxCxCxCxAxC
TCGAGAGAGAGAGAAGAGGAAGAGTCTCAGCTGAAGCTT...CTCTGACTCTTCTCTCTCTCTCTCGATC

TxGxGxGxGxGxG (ciąg zbioru S(A1) kodujący 1 graf)
 CxAxAxAxAxAxA (ciąg zbioru S(A1) kodujący 2 graf)
  GxGxGxGxGxGxA (ciąg zbioru S(A1) kodujący 1 graf)
   AxAxAxAxAxAxG (ciąg zbioru S(A1) kodujący 2 graf)
    GxGxGxGxGxAxA (ciąg zbioru S(A1) kodujący 1 graf)
     ...
TxGxGxGxGxGA  (ciąg zbioru S(A2) kodujący 1 graf)
 CxAxAxAxAxAG  (ciąg zbioru S(A2) kodujący 2 graf)
  GxGxGxGxGxGA  (ciąg zbioru S(A2) kodujący 1 graf)
   AxAxAxAxAxAA  (ciąg zbioru S(A2) kodujący 2 graf)
    GxGxGxGxGxAG  (ciąg zbioru S(A2) kodujący 1 graf)
     AxAxAxAxAxGA  (ciąg zbioru S(A2) kodujący 2 graf)
      ...
                  (ciąg zbioru S(A2) kodujący 1 graf:) CxCxCxCxCxAT
                   (ciąg zbioru S(A2) kodujący 2 graf:) TxTxTxTxGxTC
```

W uogólnionym superciągu kolejne litery nieparzyste to odpowiedniki elementów ścieżki Hamiltona w pierwszym grafie, litery parzyste - w drugim. Odpowiedniki te łatwo odtworzyć na ciągi typu Alternating, rozdzielając litery z {A, C, G, T} (odpowiednio parzyste lub nieparzyste) znakami X, aby otrzymać dwa superciągi składowe z superciągu uogólnionego.

Obydwa składowe superciągi budowane dla elementów S_{A1} mają taką samą usystematyzowaną strukturę jak opisano już dla problemu chipu Gapped z błędami pozytywnymi:

l_1 elementów "startowych", poczynając od $TxGxGxGxGxGxG$ w pierwszym superciągu przykładu i $CxAxAxAxAxAxA$ w drugim.

element będący odpowiednikiem s - odpowiednio: $AxAxGxAxAxTxT$ oraz $GxGxAxGxGxCxC$

$l_1 - 1$ elementów odpowiadających łukowi wychodzącemu z s

inne elementy reprezentujące wierzchołki i łuki w grafie

$l_1 - 1$ elementów odpowiadających łukowi wchodzącemu do t

element będący odpowiednikiem wierzchołka t

l_1 elementów "końcowych" kończąc na $TxTxTxTxTxGxT$ w pierwszym superciągu składowym oraz $CxCxCxCxCxAxC$ w drugim.

Superciągi składowe zbudowane z precyzyjnie określonej liczby elementów z S_{A1} istnieją tylko, jeżeli podciągi z tego zbioru nakładają się maksymalnie, wtedy i tylko wtedy gdy istnieje przynajmniej jedna ścieżka Hamiltona w każdym grafie instancji wyjściowej. Na uogólniony superciąg będący ich złożeniem nakłada się precyzyjnie określona liczba elementów S_{A2}. Jeżeli grafy mają więcej niż jedną ścieżkę Hamiltona, prowadzi to do możliwości zbudowania większej liczby superciągów składowych, a co za tym idzie istnieje wtedy więcej uogólnionych superciągów w danej instancji problemu qASBH-ped. Jeżeli choć jeden z grafów nie będzie mieć ścieżki Hamiltona, jeden z superciągów składowych nie będzie mógł być utworzony z określonej liczby elementów podzbioru S_{A1}, aby mieć wymaganą długość (przy konieczności maksymalnych nałożeń). Przez to nie będzie możliwe złożenie go z drugim superciągiem w jeden uogólniony superciąg, aby ten miał wymaganą długość. Długość ta to $l_1(|V|+2)+|V|+2$ znaków alfabetu {A, C, G, T}, co stanowi też rozwiązanie problemu qASBH-ped. □

Twierdzenie 6.4.2. Problem ASBH-pes (w wersji przeszukiwania) jest silnie NP-trudny.

Dowód

Jak w poprzednim rozdziale, uzasadnienie powyższego twierdzenia wynika z faktu, że gdyby dla problemu przeszukiwania ASBH-pes istniał algorytm wielomianowy, to można by go wykorzystać do rozwiązania w wielomianowym czasie problemu quasiASBH-ped [26]. Byłoby tak dlatego, że zbiór wszystkich instancji problemu przeszukiwania stanowi zbiór wszystkich instancji z odpowiedzią TAK problemu qASBH-ped. Z tego powodu można by użyć tego algorytmu dla każdej instancji problemu qASBH-ped. Jeżeli w wielomianowym czasie algorytm nie znajdzie rozwiązania, odpowiedź dla qASBH-ped brzmi NIE, TAK w przeciwnym wypadku, tj. jeśli w wielomianowym czasie algorytm znajdzie rozwiązanie. W świetle teorii złożoności oraz założenia, że $P \neq NP$ nie jest to możliwe. □

6.5 Algorytm dla chipu typu Alternating

6.5.1 Dane wejściowe

Celem algorytmu jest rekonstrukcja sekwencji DNA o znanej długości n ze spektrum DNA otrzymanego przy użyciu nieklasycznego chipu typu Alternating. Opracowany algorytm radzi sobie z występowaniem wszystkich rodzajów błędów hybrydyzacji: błędów pozytywnych oraz błędów negatywnych. Te drugie wynikać mogą zarówno z powodu błędów procesu odczytu chipu, a także z powodu powtórzeń podciągów w sekwencji DNA. Dane wejściowe algorytmu są wymienione na poniższej liście:

1) spektrum DNA zawierające elementy pochodzące z obu połówek chipu typu Alternating, składające się ze zbiorów S_{A1} oraz S_{A2};
2) długość n badanego fragmentu DNA;
3) parametr k określający długość oligonukleotydów chipu.;
4) informacja o elemencie ze zbioru S_{A1} oraz S_{A2}, które hybrydyzowały z pierwszymi dwoma podłańcuchami DNA o długości $2k-1$ nukleotydów każdy. Te dwa ciągi / wzory sond, określają więc pierwsze $2k$ nukleotydów badanego DNA (przy maksymalnym nałożeniu na siebie na $l_1 - 1$ znakach).

Parametry n oraz k pozwalają obliczyć dodatkowe wartości liczbowe, niezbędne dla dalszej pracy algorytmu. Parametr k pozwala określić długość oligonukleotydów w sondach klasycznych i nieklasycznych (6.1). Mając daną długość DNA n, możliwe jest określenie hipotetycznej liczby elementów obu zbiorów, którą zawierałoby spektrum idealne (6.2). Gdyby możliwe było wykluczenie któregokolwiek z rodzajów błędów hybrydyzacji, możliwe stałoby się precyzyjne określenie liczby błędów w spektrum danym. W innym wypadku algorytm określa tę liczbę w sposób przybliżony. Jak zostanie wyjaśnione w dalszej części rozdziału, jest to wymagane, aby ograniczyć przestrzeń rozwiązań, którą algorytm przeszukuje.

Spektrum DNA jest podzielone na dwa zbiory: zbiór S_{A1} oraz zbiór S_{A2}. Oba zawierają elementy zbudowane nad wspólnym 5-literowym alfabetem, różnią się jednak długością elementów. Zbiór S_{A1} jest używany do budowy grafu skierowanego. Zasada działania procedury tworzenia takiego grafu jest następująca:

1) wszystkie elementy zbioru S_{A1} są traktowane jako wierzchołki grafu, opisane nad alfabetem 5-literowym, każdy o długości l_1;

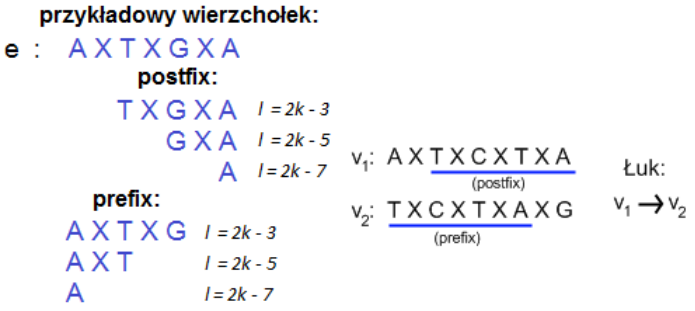
2) łuki w grafie tworzone są na bazie nakładania na siebie postfixu i prefixu dwóch wierzchołków na długości od 1 do $l_1 - 2$ (z krokiem co 2 znaki).

Innymi słowy można nakładania to opisać jako takie, w którym nie obowiązuje reguła nakładania ciągów typu Alternating, tzn. litera X nakłada się tutaj tylko na samą siebie. Ta odmienna zasada obowiązuje tylko w czasie tworzenia grafu.
W dalszej części rozdziału określenia "ciąg" i "wierzchołek" (a także "następnik") używane będą zamiennie, ponieważ każdy wierzchołek grafu jest opisany przez pewien ustalony ciąg znaków, tj. ciąg typu Alternating, ze zbioru S_{A1}.

Postfixem o długości l_{post} ciągu e nazywamy podciąg jego ostatnich l_{post} liter, przy czym długość maksymalna postfixu jest równa $l_{post}(max) = l_{post} - 2$ ostatnich liter, minimalna długość to 1 (ostatnia litera), z krokiem co 2 litery. Na przykład, postfixem maksymalnym o długości 5 wierzchołka opisanego jako $AXTXGXA$, jest podciąg $TXGXA$, kolejny, krótszy postfix o długości 3 to w przykładzie GXA, postfix minimalny to A.

Prefixem o długości l_{pref} ciągu e nazywamy podciąg jego początkowych l_{pref} liter, przy czym długość maksymalna prefixu jest równa $l_{pref}(max) = l_{pref} - 2$ początkowych liter, minimalna długość to 1 (pierwsza litera) z krokiem co 2 litery. Na przykład, prefixem maksymalnym o długości 5 wierzchołka opisanego jako $AXTXGXA$, jest podciąg $AXTXG$, kolejny krótszy prefix o długości 3 to AXG, prefix minimalny to A.

Dwa ciągi e_1 i e_2 nakładają się na siebie na długości r, jeżeli postfix ciągu e_1 o długości r jest identyczny z prefixem e_2 o tej samej długości. Podane tutaj definicje ilustruje Rysunek 6.2:



Rysunek 6.2: Prefix, postfix oraz tworzenie łuku w grafie

Dla każdego wierzchołka e ze zbioru S_{A1} algorytm testuje występowanie wspólnych postfixów / prefixów z każdym innym wierzchołkiem z S_{A1}. W efekcie powstaje graf skierowany, w którym połączone są ze sobą wszystkie wierzchołki. Z każdym łukiem

związana jest pewna wartość "wagi". Zależna jest ona od liczby znaków, na których nakładają się na siebie dwa ciągi / wierzchołki. Algorytm przyjmuje, że waga połączenia jest równa 1, jeżeli dwa wierzchołki nałożyły się na $l_1 - 2$ znakach (czyli maksymalnie), 2 jeżeli było to $l_1 - 4$ liter, itd. Oczywiście w procesie rekonstrukcji DNA, czyli poszukiwania ścieżki w grafie, preferowane są łuki z jak najniższymi wagami.

Przyjęto założenie, że błędy negatywne polegające na zagubieniu pewnych elementów wymaganych do rekonstrukcji DNA, mają niskie prawdopodobieństwo wystąpienia obok siebie tworząc "wyrwę" w elementach składowych o długości większej niż k nukleotydów z krokiem co dwa. Na przykład k kolejnych nieparzystych lub parzystych nukleotydów w sekwencji reprezentowanych przez elementy z S_{A1} nie będzie możliwych do rekonstrukcji, jeżeli wymienione tu k kolejnych elementów S_{A1} zostało zagubionych w wyniku błędów negatywnych. Taka sytuacja uniemożliwi rekonstrukcję prawdziwego badanego DNA, lecz jest bardzo mało prawdopodobna, szczególnie w przypadku przyjęcia najniższej używanej wartości parametru k równej 8.

Druga część spektrum to zbiór S_{A2}. Jest on traktowany jako lista, która posłuży do weryfikacji kolejno wybieranych do ścieżki wierzchołków ze zbioru S_{A1}.

Ostatnią ważną daną wejściową jest wiedza o pierwszych $2k$ znakach badanego łańcucha DNA. Jest tutaj pewna istotna różnica w porównaniu z poprzednim zaproponowanym algorytmem (tj. dla chipu Gapped). Znajomość tylu liter DNA pozwala na wiedzę nie tylko o pierwszym wierzchołku e_{s1} grafu, lecz także o drugim wierzchołku e_{s2}, gdzie oba mają długość $l_1 = 2k - 1$. Oba są traktowane jako wierzchołki startowe, co zostanie dokładniej wyjaśnione w dalszej części rozdziału. Analizowane DNA hybrydyzowało za pomocą odpowiednio k pierwszych nieparzystych nukleotydów, oraz k pierwszych parzystych nukleotydów do sond opisanych wzorem pierwszego i drugiego wierzchołka danego. Wierzchołki te "nakładają" się na siebie w ten sposób, że tam gdzie pierwszy ma w swoim opisie nukleotydy z {A, C, G, T}, drugi ma znak X i odwrotnie. Pojawia się tutaj zasadniczy problem dla dalszego przeszukiwania - jak dopasować wierzchołki do siebie aby rekonstruować badane DNA, skoro do symbolu X pasuje jakikolwiek naturalny nukleotyd w procesie nakładania.

Dzięki znajomości dwóch pierwszych wierzchołków, będących odpowiednikami k pierwszych nieparzystych nukleotydów i k pierwszych parzystych nukleotydów, utworzony ze zbioru S_{A1} graf ma dwa wierzchołki "startowe". Od nich ścieżki są rekonstruowane krokowo, co jeden nowy wierzchołek raz dla części "nieparzystej" rekonstruowanej sekwencji, następnie dla części "parzystej", itd. Za zachowanie spójności rozwiązania, tj. zapewnienie, że nukleotydy nieparzyste i parzyste rekonstruują wciąż to samo badane DNA, odpowiada weryfikacja następników za pomocą elementów zbioru S_{A2}. W tej sytuacji mamy do czynienia z jednym grafem, dla którego symultanicznie szukamy dwóch ścieżek, przy czym algorytm zachowuje długość obu ścieżek (tj. liczbę rekonstruowanych nieparzystych i parzystych znaków DNA) w każdym kroku poprzez wstrzymanie rekonstrukcji

jednej z nich, jeśli jest to wymagane. Przykładowo, jeśli dołożony fragment rozszerzył jedną ścieżkę poprzez np. mniejsze niż maksymalne nałożenie z ostatnim wierzchołkiem, jej dalsze rozszerzanie jest wstrzymywane do czasu, aż druga rekonstruowana ścieżka "dogoni" pierwszą (z uwagi na liczbę zrekonstruowanych dotąd nukleotydów). Ma to miejsce w efekcie dodania do niej odpowiedniej liczby pasujących wierzchołków. W każdym kroku następuje weryfikacja następnika elementem ze zbioru S_{A2} tak, aby w każdej chwili rekonstruowano poprawne rozwiązanie, tj. spełniające warunki postawionego problemu. Ten mechanizm zostanie dokładnie opisany w następnym podrozdziale, poświęconym konstrukcji i zasadzie działania algorytmu.

6.5.2 Zasada działania algorytmu

Opracowany został algorytm dokładny, który stara się przeszukać całą przestrzeń rozwiązań. Z uwagi na wbudowane parametry ogranicza on poszukiwania do pewnej jej części. Jest to spowodowane ograniczeniem czasu obliczeń oraz innymi parametrami, które zostaną dalej wyjaśnione. Algorytm poza najważniejszymi danymi wejściowymi, biorącymi swój początek w eksperymencie hybrydyzacyjnym, posiada też inne parametry regulujące jego pracę. Możliwe jest takie ich ustawienie, aby przeszukiwał on całą dostępną przestrzeń rozwiązań. Z uwagi na jej ogromny rozmiar, algorytm ogranicza jednak przeszukiwanie dla pewnego dostępnego czasu obliczeń, liczby wewnętrznych iteracji czy znalezionych rozwiązań niejednoznacznych w wyznaczonym czasie. Parametry te zostaną opisane w części dotyczącej warunków zatrzymania przeszukiwania.

Algorytm poszukuje dwóch ścieżek w grafie, pozwalających zrekonstruować DNA o ustalonej długości n. Pierwsza rekonstruowana ścieżka algorytmu, z wierzchołkiem startowym e_{s1} odpowiada za kolejność nukleotydów nieparzystych DNA P_{odd}, druga rekonstruowana ścieżka P_{even} rozpoczynająca się w wierzchołka e_{s2} odpowiada za kolejność nukleotydów parzystych w poszukanej sekwencji. Wierzchołek grafu może znaleźć się na którejkolwiek ścieżce tylko raz, tj. jeśli dany wierzchołek został użyty w ścieżce P_{even}, od tego momentu nie może on zostać dodany do ścieżki P_{odd}. Algorytm dokłada kolejne wierzchołki krokowo, raz do jednej, następnie do drugiej ścieżki. W momencie, w którym dwie ścieżki osiągną długość odpowiednią, aby po nałożeniu ich na siebie odtworzyć całą sekwencję DNA, takie rozwiązanie jest dodawane do listy rozwiązań, po sprawdzeniu odpowiednich warunków poprawności opisanych dalej. Algorytm następnie wycofuje się do wcześniej użytych wierzchołków, sprawdzając ich inne osiągalne następniki. Ogólny schemat działania można przedstawić w formie następującego Algorytmu 3 zapisanego jako pseudokod odpowiednich poleceń.

Powyższy pseudokod przedstawia główną pętlę algorytmu. Jest ona identyczna jak w przypadku poprzedniego algorytmu opracowanego dla chipu typu Gapped. Zostanie

Algorithm 3 Pseudokod głównej pętli algorytmu dla chipu typu Alternating.

```
1: while (Time < Limit AND s_space ≠ empty AND Stop = FALSE) do
2:     Candidates ← addOutgoingVerticesToList(Current_Vertex)
3:     Success_Flag ← addNewVertexToSolution(Candidates);
4:     if (Success_Flag = FALSE) then
5:         if (Path_Length = MaxValue) then
6:             Ok = testSolution(Solution);
7:             if (Ok = TRUE) then
8:                 addToSolutionList(Solution)
9:                 reverseSteps(Solution)
10:            else
11:                reverseSteps(Solution)
12:            end if
13:        else
14:            reverseSteps(Solution)
15:        end if
16:    end if
17: end while
```

jednak tutaj omówiona, w celu przybliżenia ogólnej zasady działania. Różnice między algorytmami zawarte są "głębiej" w kodzie, przede wszystkim w procesie doboru kolejnego wierzchołka oraz w części odpowiedzialnej za weryfikację kolejnych elementów ścieżek. Zostaną one dokładnie omówione w dalszej części tego podrozdziału.

W każdym kroku pętli głównej do aktualnie rozpatrywanej ścieżki dodawany jest nowy wierzchołek. Na tym etapie nie ma jeszcze znaczenia, czy jest to ścieżka P_{odd} czy P_{even}. Na początku każdego kroku pętli, w linii 1 algorytm porównuje czas, który upłynął od rozpoczęcia pracy, z limitem podanym jako całościowy dostępny czas przeszukiwania. Praca przerywana jest także po przeszukaniu całej przestrzeni rozwiązań. Zmienna *Stop* w pseudokodzie odpowiada za inne warunki, które mogą być nałożone na algorytm, jak na przykład limit znalezionych rozwiązań czy limit nieprawidłowych rekonstrukcji o długości n niespełniających warunków weryfikacji sprawdzanych w linii 6. Lista wszystkich warunków zatrzymania pętli to:

1) sprawdzono całą przestrzeń rozwiązań;

2) osiągnięto limit czasu na przeszukiwanie;

3) osiągnięto limit rozwiązań dodanych do listy rozwiązań (limit rozwiązań niejednoznacznych);

4) osiągnięto limit rekonstrukcji par ścieżek o zadanej długości, które następnie zostały odrzucone przez procedurę weryfikacji rozwiązań.

Procedura budowania listy kandydatów *Candidates*, czyli następników aktualnie odwiedzonego wierzchołka jest wywoływana w linii 2. W tej części kodu następuje także weryfikacja potencjalnych nowych następników przy użyciu elementów zbioru S_{A2}, w celu zapewnienia odpowiedniego dopasowania nukleotydów nieparzystych i parzystych rekonstruowanych przez obie ścieżki.

Po przygotowaniu listy potencjalnych następników, kolejna procedura algorytmu odpowiada za próbę dodania nowego wierzchołka (następnika) do ścieżki (linia 3). W tej części algorytm sprawdza warunki, które wierzchołek z listy kandydatów musi spełnić, aby mógł być dodany jako następnik. Następnie sprawdzany jest zestaw warunków mających na celu określenie dalszego toku pracy. Jeżeli wierzchołek został dodany poprawnie, iteracja pętli głównej kończy się, po czym o ile stan warunków pętli *while* nie powoduje zakończenia poszukiwań, algorytm buduje nową listę kandydatów na następników, od ostatniego wierzchołka "parzystego" jeśli poprzednio dodano "nieparzysty" i odwrotnie. Jeżeli nowy wierzchołek nie został dodany w linii 3, sprawdzane są tego powody (linia 4 i kolejne). Jeżeli algorytm wykryje, że ostatnio dodany wierzchołek kończy ścieżkę o wielkości umożliwiającej odtworzenie DNA o odpowiedniej długości, przy czym druga ścieżka taką wielkość już osiągnęła w poprzednim kroku, sprawdzane jest, czy ścieżki spełniają warunki aby zostać zaakceptowane. Jeśli tak, odtwarzany jest z nich uogólniony superciąg (linia 6). Algorytm dodaje takie DNA w formie superciągu do listy rozwiązań, po czym kontynuuje główną pętlę, w kolejnych iteracjach dodając inne wierzchołki po wcześniejszych wycofaniu się odpowiednią liczbę kroków. Po dodaniu nowego rozwiązania, a także w przypadku niemożności zaakceptowania jakiegokolwiek nowego wierzchołka z listy kandydatów, algorytm w sekcjach w liniach 9, 11 i 14 przechodzi do wydzielonej procedury mającej na celu powrót do wierzchołka, z którego możliwy był krok prowadzący do innych nieodwiedzonych jeszcze następników. Procedura ta próbuje dodać kolejny wierzchołek do listy, jeśli jest to możliwe. Jeśli się to uda, następuje kolejne wykonanie pętli głównej algorytmu. Jeśli nie, oznacza to wyczerpanie się przestrzeni rozwiązań, którą algorytm przeszukiwał. Jest to koniec pracy głównej pętli algorytmu, po czym następuje przejście do sekcji wygenerowania wyników rekonstrukcji DNA.

Następna procedura całego algorytmu, która zostanie przedstawiona, jest odpowiedzialna za tworzenie zbioru kandydatów na następników aktualnie odwiedzonego wierzchołka. Ilustruje ją poniższy Algorytm 4:

Procedura ta jest podobna do przedstawionej dla chipu typu Gapped, ma jednak bardzo istotną różnicę, o której będzie mowa dalej. W linii 1 zawarta jest główna pętla, która sprawdza wszystkie istniejące połączenia wychodzące z aktualnie dodanego do

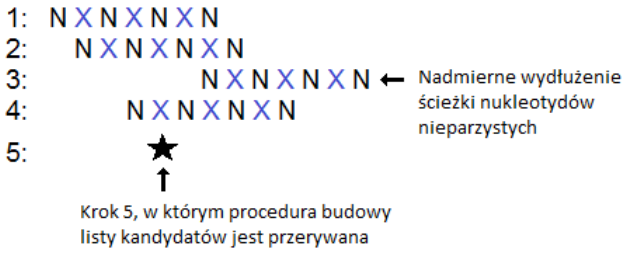
Algorithm 4 Pseudokod procedury tworzenia listy kandydatów na następników.

```
1: for Vertex IN AllOutgoingVerticesSet do
2:     if (current_path » last_path) then
3:         break;
4:     else
5:         if (checkIfUnused(vertex) = TRUE) then
6:             Verified = verifyVertex(vertex, S_A2);
7:             if (Verified = TRUE) then
8:                 addVertexAsCandidate(vertex, Candidates)
9:             end if
10:        end if
11:    end if
12: end for
```

ścieżki wierzchołka. Sprawdzane są one w kolejności od największego do najmniejszego nałożenia, tj. najpierw algorytm sprawdza następniki o maksymalnym nałożeniu równym $l_1 - 2$ oraz wadze $w_{overlap} = 1$, następnie te nakładające się na $l_1 - 4$ znakach o wadze $w_{overlap} = 2$, itd.

W linii 2 określony jest warunek, który odróżnia tę procedurę od opisanej w poprzednim rozdziale pracy. Jeśli chodzi o tę różnicę, należy pamiętać o dwóch rzeczach. Po pierwsze omawiany algorytm pracuje na dwóch ścieżkach, w kolejnych krokach pętli głównej powiększając raz jedną, raz drugą. Po drugie, w wyniku pojawienia się błędów negatywnych w zbiorze S_{A1}, konieczne stanie się wybranie w pewnym momencie wierzchołków o nałożeniu mniejszym niż maksymalne. Budowanie dwóch ścieżek w jednym grafie może też powodować, że jeżeli pewien wierzchołek z S_{A1} powtórzy się w DNA na nukleotydach parzystych i nieparzystych (zgodnie z regułą kodowania Alternating), powstanie bardzo specyficzny rodzaj błędu negatywnego wynikającego z powtórzeń. Wtedy również będzie potrzeba wykorzystania nakładania mniejszego niż maksymalne (tj. mniejsze niż na $l_1 - 2$ znakach). W pewnym momencie jedna ze ścieżek może "wybiec" do przodu w stosunku do drugiej. Kiedy taka sytuacja zostaje wykryta, procedura konstruowania listy następników dla takiej ścieżki jest wstrzymywana. Do pętli głównej wysyłana jest informacja, aby pomimo tego, że lista kandydatów nie została utworzona (a co za tym idzie żaden nowy następnik nie zostanie dodany) nie należy wykonywać powrotu do poprzednich wierzchołków, ale pominąć cały aktualny krok (w nadmiernie rozszerzonej ścieżce) i przejść do kroku następnego - tj. powiększania ścieżki krótszej. Ilustruje to poniższy przykład na rysunku 6.3:

Na powyższym rysunku przedstawiono sytuację, w której w trzecim kroku algorytm użył wierzchołka z minimalnym nałożeniem, czyli takim, które maksymalnie rozszerza



Rysunek 6.3: Przykład nadmiernego poszerzenia jednej ze ścieżek

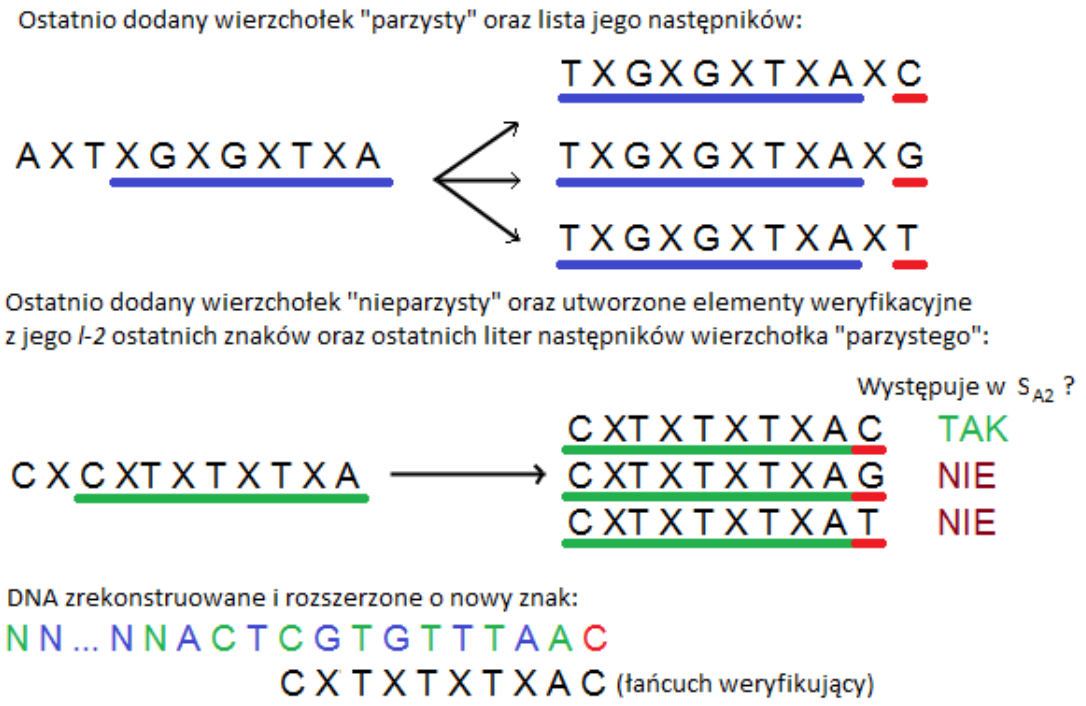
ścieżkę. W kroku czwartym algorytm dodaje wierzchołek "parzysty", tj. z maksymalnym dopuszczalnym nałożeniem. Opisywana sytuacja zatrzymania wykonywania listy kandydatów ma miejsce w kroku piątym. Algorytm wykrywa, że ścieżka nieparzysta jest już nadmiernie rozszerzona w stosunku do parzystej. Wykonanie procedury jest zatrzymywane, algorytm zaś dostaje polecenie przejścia od razu do drugiej ścieżki. Jeśli ten krok nadal nie rozszerzy ścieżki parzystej ponad nieparzystą (w przykładzie), także następny krok dla ścieżki dłuższej zostanie przerwany w ten sam sposób.

Jeśli opisane przerwanie nie nastąpi, reszta procedury wykonuje się podobnie jak w przypadku poprzedniego algorytmu. W linii 5 algorytmu 4 sprawdzana jest informacja, czy wierzchołek brany pod uwagę jako następnik nie został jeszcze użyty. Linia 6 omawianego pseudokodu odpowiada za wykonanie procedury weryfikacji potencjalnego następnika. Nie ma tutaj warunku na odpowiednią długość rekonstruowanych ścieżek - procedura ta działa od pierwszego do ostatniego kroku algorytmu, ponieważ zakładając znajomości dwóch pierwszych wierzchołków (startowych) jest ona od razu możliwa do przeprowadzenia. Jeżeli weryfikacja się powiedzie, co jest sprawdzane w warunku z linii 7, wierzchołek jest dodawany do listy potencjalnych kandydatów (linia 8). Pętla trwa tak długo, aż sprawdzone (zweryfikowane) zostaną wszystkie następniki aktualnie dodanego wierzchołka.

Kolejną procedurą, która zostanie teraz omówiona jest weryfikacja następnika, zanim zostanie on dodany do listy kandydatów. Jest ona niezbędna w omawianym podejściu, ponieważ to ona zapewnia zgodność rekonstrukcji nukleotydów nieparzystych i parzystych dla poszukiwanej sekwencji. Zanim zostanie jednak opisany jej pseudokod, przedstawione zostaną przypadki weryfikacji, które mogą wystąpić w czasie pracy algorytmu. Ułatwi to zrozumienie pseudokodu samej procedury weryfikacji, który zostanie przedstawiony za przykładami.

Pierwszy przypadek weryfikacji, w ramach którego objaśniony zostanie ten mechanizm, pokazano na rysunku 6.4:

Wierzchołek podkreślony na zielono: $CXCXTXTXTXA$ został dodany krok wcześniej, do ścieżki P_{odd} (nieparzystej). W przykładzie, algorytm znajduje się w stanie, w

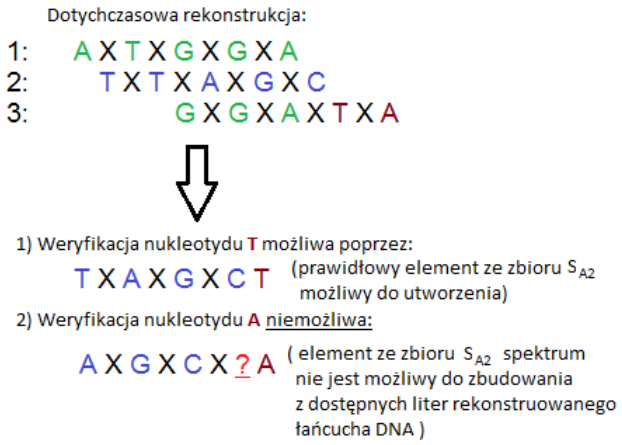


Rysunek 6.4: Alternating Chip - przykładowy schemat weryfikacji

którym musi zweryfikować 3 potencjalne następniki dla ostatnio dodanego do ścieżki P_{even} wierzchołka $AXTXGXGXTXA$. Są to wierzchołki:
$TXGXGXTXAXC$, $TXGXGXTXAXG$ oraz $TXGXGXTXAXT$.

Wszystkie trzy nakładają się maksymalnie na ostatnio dodany. Wybranie któregokolwiek z nich jako następnika rozszerzy ścieżkę parzystą o jeden dodatkowy nukleotyd z {A, C, G, T}. Weryfikacja każdego potencjalnego następnika polega na stworzeniu elementu, o budowie pasującej do zbioru S_{A2} spektrum oraz sprawdzeniu jego obecności w S_{A2}. Element weryfikacyjny składa się z testowanego, ostatniego nukleotydu następnika oraz postfixu ostatnio dodanego wierzchołka drugiej ścieżki (w tym wypadku ze ścieżki nieparzystej, gdyż następniki dotyczą wierzchołków ścieżki parzystej). Do listy potencjalnych następników zostaną dodane tylko te wierzchołki, których elementy weryfikacyjne istnieją w zbiorze S_{A2} spektrum. Podejście to, w całości wynikające ze specyfiki chipu typu Alternating, pełni tutaj dwojaką funkcję. Po pierwsze, tak jak realizowano weryfikację poprzednim algorytmie, ogranicza ono tutaj liczbę możliwych następników wierzchołka. Ważniejszą funkcją weryfikacji jest jednak zapewnienie "połączenia" między rekonstruowanymi ścieżkami, poprzez zapewnienie, że z obydwu rekonstruowane jest DNA spełniające kryteria rozwiązania.

Rysunek 6.5 przedstawia weryfikację elementu, w którym nie wszystkie nowe nukleotydy mogą zostać zweryfikowane.

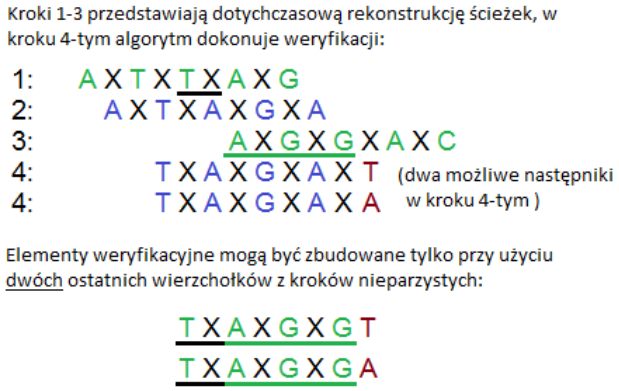


RYSUNEK 6.5: Alternating Chip - schemat niepełnej weryfikacji

Przedstawiony na rysunku przykład prezentuje sytuację, w której nowy potencjalny następnik nakłada się na prefixie krótszym niż maksymalny możliwy (na rysunku jego długość to $l_{pref} = 2k - 5$ gdzie $k = 5$, przy maksymalnym teoretycznym nałożeniu mającym długość $pref_{max} = 2k - 3$). Weryfikacja nukleotydu T (pierwszy dodany) jest możliwa. Niemożliwa jest jednak weryfikacja ostatniego nukleotydu (tj. A), ponieważ ostatni element drugiej ścieżki (dodany w poprzednim kroku pętli głównej) nie rozszerzył tejże ścieżki do długości niezbędnej do zbudowania pełnego elementu weryfikacyjnego. W takim wypadku następnik zostanie dodany wtedy, gdy element $TXAXGXCXT$ zostanie odnaleziony w spektrum w zbiorze S_{A2}. Pojawia się tutaj pytanie co się stanie, jeżeli niezweryfikowany nukleotyd nie jest prawidłowy z punktu widzenia optymalnej sekwencji DNA (tj. cały wierzchołek dodający go tutaj taki nie jest). Możliwe są dwie sytuacje. W pierwszej, bardziej prawdopodobnej, nieprawidłowy nukleotyd sam uniemożliwi pewną weryfikację w dalszych krokach algorytmu, co spowoduje, że w pewnym momencie algorytm wycofa się do tego miejsca i spróbuje zastąpić nieprawidłowy wierzchołek innym. Mniej prawdopodobna choć wciąż możliwa jest sytuacja powstania sekwencji DNA, która spełnia warunki rozwiązania dla problemu, lecz będzie powodowała powstanie rozwiązania niejednoznacznego po zakończeniu przez algorytm przeszukiwań przestrzeni rozwiązań. Także z tego powodu końcowe rozwiązanie, czyli zrekonstruowane z dwóch odpowiedniej długości ścieżek DNA, jest ponownie weryfikowane elementami ze zbioru S_{A2}, zanim zostanie dodane do zbioru rozwiązań dopuszczalnych. Będzie o tym dalej mowa przy omawianiu odpowiedniej procedury algorytmu.

Kolejny przykład, na rysunku 6.6 przedstawia sytuację, w której weryfikacja wymaga dwóch ostatnich wierzchołków drugiej ścieżki (w stosunku do aktualnie przetwarzanej).

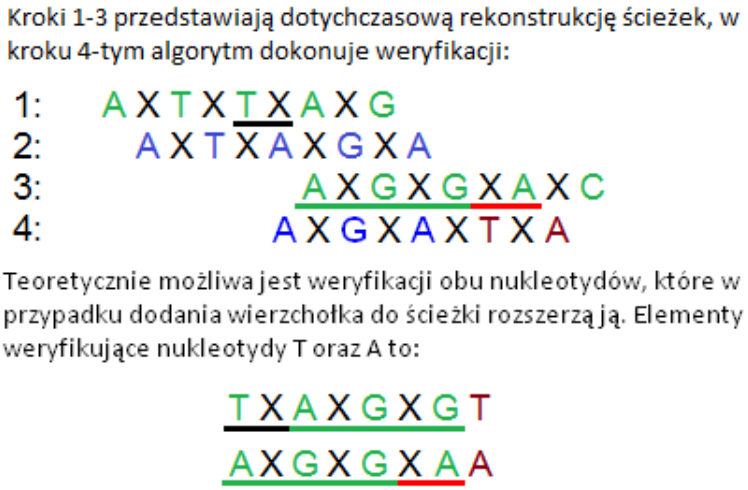
Przykład na rysunku pokazuje stan, w którym algorytm dodał już kilka wierzchołków do ścieżek (kroki 1 - 3). Ponieważ jednak wcześniej nastąpiło przedłużenie jednej z nich



RYSUNEK 6.6: Alternating Chip - schemat weryfikacji dwoma wierzchołkami

z użyciem nałożenia mniejszego niż maksymalne (krok 3 na rysunku), aby zweryfikować potencjalne następniki aktualnego wierzchołka (w kroku 4), należy złożyć element weryfikacyjny z więcej niż jednego elementu składowego, tj. wierzchołka poprzedniej ścieżki. Na rysunku na czarno podkreślony został fragment, który został pobrany z przedostatnio dodanego wierzchołka poprzedniej ścieżki.

Ostatni przykład weryfikacji pokazuje pewien dylemat występujący wtedy, kiedy teoretycznie wszystkie nowe nukleotydy dodawane przez testowany następnik mogą zostać zweryfikowane. Pokazany jest on na rysunku 6.7.



RYSUNEK 6.7: Alternating Chip - schemat weryfikacji z możliwością testowania wszystkich nowych nukleotydów

W przykładzie nowy następnik w przypadku dodania go do ścieżki, doda do niej dwa nowe nukleotydy: T oraz A. Oba mogą zostać zweryfikowane, ponieważ w przeciwieństwie do przykładu z rysunku 6.5 poprzednio rozpatrywana przez algorytm ścieżka ma

odpowiednią długość. Zastosowanie takiej dokładnej weryfikacji, tj. wszystkich dodatkowych nukleotydów jeśli jest to możliwe, jest uzależnione w algorytmie od tego, jakie rodzaje błędów występują w spektrum. Jeżeli występują w spektrum błędy negatywne, weryfikowanie wszystkich nowych nukleotydów może doprowadzić do sytuacji, w której prawidłowe (z punktu widzenia prawidłowego rozwiązania) w danym miejscu wierzchołki ze zbioru S_{A1} zostaną odrzucone tylko dlatego, że część "ich" elementów weryfikacyjnych nie będzie obecna w S_{A2}. Dlatego taka pełna weryfikacja odbywa się tylko w przypadku braku błędów negatywnych w spektrum. Weryfikacja jest wtedy dokładniejsza, niestety nie można takiej sytuacji zakładać po realnym eksperymencie w metodzie ASBH, gdzie spektrum może zawierać błędy negatywne.

Drugi problem wynika pośrednio z sytuacji przedstawionej na rysunku 6.5. Nawet przy braku błędów negatywnych wynikających z zagubienia elementów w spektrum, może wystąpić sytuacja, dla której niektórych dodatkowych dla ścieżki nukleotydów nie da się zweryfikować, przy czym z punktu widzenia całego badanego DNA nie będą one na prawidłowych pozycjach. Mowa tu o ostatnich nukleotydach nowego wierzchołka, których weryfikacja jest niemożliwa, ponieważ poprzednio rozszerzona ścieżka wciąż jest zbyt krótka, aby utworzyć element weryfikacyjny. Jeśli taki wierzchołek zostanie dodany do ścieżki (ponieważ była możliwa weryfikacja pierwszego nowego dla ścieżki nukleotydu i tylko jego), dalsze, nieprawidłowe nukleotydy wpłyną na prawidłowość dalszych faz weryfikacji. W tym wypadku elementy weryfikacyjne z ich wykorzystaniem (tj. nieprawidłowych nukleotydów) również będą nieprawidłowe i prawdopodobnie nie wystąpią w zbiorze S_{A2}. W tej sytuacji algorytm w pewnym momencie wróci do punktu, w którym zamieni nieprawidłowy wierzchołek na prawidłowy, lecz odbędzie się to kosztem zasobów zużytych na przeszukanie części przestrzeni rozwiązań. W konstrukcji algorytmu przyjęto, że weryfikacja wszystkich nowych nukleotydów odbywa się tylko w teoretycznym przypadku nie występowania błędów negatywnych. Algorytm w realistycznym trybie pracy (zakładającym istnienie wszystkich rodzajów błędów w spektrum), weryfikuje zawsze tylko jeden nukleotyd następnika - ten, który jako pierwszy rozszerza ścieżkę ponad jej aktualną długość.

Teraz można przejść do opisu procedury algorytmu 5 odpowiedzialnej za weryfikację.

W pierwszym kroku w linii 1 algorytm sprawdza, czy aktualnie weryfikowany wierzchołek rozszerzy ścieżkę o więcej nukleotydów, niż posiada ścieżka poprzednia. Na przykład, czy nowy wierzchołek w ścieżce nieparzystej, która jest właśnie rozszerzana, w razie dodania go jako następnika rozszerzy tą ścieżkę nieparzystą bardziej, niż wynosi długość ścieżki poprzedniej - parzystej. Jeśli nie, oznacza to, że teoretycznie możliwe jest sprawdzenie wszystkich nukleotydów, które rozszerzają ścieżkę, ponieważ dla każdego da się zbudować element weryfikacyjny. Drugim warunkiem (linia 5) jest jednak

Algorithm 5 Weryfikacja nowego następnika.

```
1: if (CurrentPath + NewVertex < PreviousPath) then
2:     if (NegativeErrors = TRUE) then
3:         VerificationVertex ← create(use_first_expanding_nucleotide)
4:         Result ← verify(NewVertex, VerificationVertex)
5:     else
6:         for every_extending_nucleotide do
7:             VerificationVertex ← create(use_expanding_nucleotide)
8:             Result ← verify(NewVertex, VerificationVertex)
9:         end for
10:    end if
11: else
12:    if (CurrentPath + NewVertex > PreviousPath) then
13:        VerificationVertex ← create(use_first_expanding_nucleotide)
14:        Result ← verify(NewVertex, VerificationVertex)
15:    end if
16: end if
17: return Result;
```

brak błędów negatywnych. Tylko wtedy nastąpi weryfikacja wszystkich nowych nukleotydów następnika, jak pokazano na rysunku 6.7. Weryfikacja następuje w liniach 3 oraz 4 kodu. Jeżeli zakłada się brak błędów negatywnych, linia 6 zawiera pętlę, która weryfikuje każdy nadmiarowy nukleotyd następnika, co ma miejsce w liniach 7 i 8.

Ostatni wariant procedury w linii 12, to przeciwieństwo warunku z linii 1 - jest on stosowany, jeżeli weryfikowany wierzchołek rozszerza ścieżkę powyżej długości ścieżki poprzedniej. W takim wypadku niemożliwe będzie sprawdzenie więcej niż pierwszego nowego nukleotydu dodawanego przez badany wierzchołek, czyli sytuacja opisana na rysunku 6.5.

Pod koniec procedura zwraca wyniki weryfikacji (linia 17). Zmienne logiczne $TRUE$ i $FALSE$ oznaczają dla algorytmu odpowiednio pozytywną lub negatywną weryfikację. W tym drugim przypadku wierzchołek nie będzie brany do listy kandydatów na następników.

Kilka procedur głównej pętli algorytmu wciąż wymaga szerszego omówienia. Procedura dodania nowego wierzchołka do ścieżki jest podobna jak w przypadku algorytmu dla chipu typu Gapped. Warunki dodania wierzchołka, które są sprawdzane, to np. ograniczenie na maksymalną długość ścieżki. Pod koniec przeszukiwania niektóre wierzchołki nakładające się na mniejszej niż maksymalna liczbie znaków, nie będą brane pod uwagę, jeśli rozszerzają ścieżkę bardziej niż potrzeba dla zrekonstruowania z jej pomocą DNA o określonej długości. Ważniejszym warunkiem jest jednak ograniczenie liczby

użytych wierzchołków nakładających się na długości $l_{overlap}$ mniejszej niż maksymalna $l_{overlap} < max_{overlap}$ gdzie $max_{overlap} = l_1 - 2$. Przestrzeń rozwiązań do przeszukania przez algorytm jest w oczywisty sposób zależna od ilości połączeń między wierzchołkami. Przy maksymalnym nałożeniu, każdy wierzchołek może mieć do czterech następników. Nałożenie o stopień mniejsze powoduje podwyższenie tego teoretycznego maksimum do 16, itd. Dlatego też istnieje pewien limit, ile razy w rekonstrukcji ścieżek można użyć takich nałożeń (tj. mniejszych niż maksymalne). Jest on całkowicie zależny od liczby błędów hybrydyzacji, dlatego w tym miejscu pojawia się problem jego obliczenia. Wpływ różnych rodzajów błędów na określanie tej wartości, a co za tym idzie na działanie całego algorytmu, podsumowują następujące wymienione przypadki:

1) Brak błędów w spektrum. Możliwa jest precyzyjna weryfikacja, tj. wiadomo, że wszystkie elementy DNA są obecne w zbiorze S_{A1}, oraz zawsze będą mogły być prawidłowo zweryfikowane przez zbiór S_{A2}. Limit wierzchołków nienakładających się maksymalnie jest równy zero. Niestety, jest to skrajnie nierealistyczny przypadek. Nawet po wyeliminowaniu jakichkolwiek nieprecyzyjności procesu hybrydyzacji i detekcji sond, pozostaje problem błędów negatywnych wynikających z powtórzeń, które są zależne jedynie od struktury badanego DNA. Wymagane byłoby tutaj bezbłędne określenie liczby powtórzeń każdego elementu, czego na chwilę obecną nie sposób wymagać.

2) W spektrum występują tylko błędy negatywne wynikające z powtórzeń. Ten rodzaj błędów negatywnych nie ma wpływu na działanie procesu weryfikacji, tj. zbiór S_{A2} może poprawnie zweryfikować każdy następnik. Błędy te mają wpływ na tworzenie ścieżek, gdyż aby rozszerzyć fragment ścieżki, gdzie dany wierzchołek nie może zostać użyty ponieważ wykorzystano go wcześniej, należy użyć następnika z mniejszym stopniem nałożenia. Liczba takich sytuacji może być jednak precyzyjnie określona przez różnicę między liczbą elementów spektrum danego, a wartościami opisującymi spektrum idealne. Jest to limit ich użycia, o którym była mowa wcześniej. Ogranicza on przestrzeń rozwiązań, nie pozwalając na przeszukiwanie obszarów, w których z całą pewnością nie ma rozwiązań dopuszczalnych dla danego spektrum. W omawianych przypadku wiadomo również, że wszystkie elementy zbioru S_{A1} muszą zostać użyte w procesie budowy dwóch ścieżek grafu.

3) Błędy negatywne ogólnego rodzaju (wynikające z powtórzeń oraz nieprecyzyjności procesu hybrydyzacji). W tym wypadku znany jest limit wykorzystania nałożeń mniejszych niż maksymalne - jest to jak napisano wyżej, różnica w ilości elementów spektrum danego i spektrum idealnym. Pojawia się tutaj problem braku precyzji w określeniu, ile dokładnie elementów zbioru S_{A1} musi zostać użytych. Znana jest

tylko minimalna liczba, równa liczności zbioru S_{A1} pomniejszonej o liczbę błędów negatywnych w zbiorze S_{A2}:

$$min\_used_{S_{A1}} = |S_{A1}| - (|S^{is}_{A2}| - |S_{A2}|)$$

gdzie $|S^{is}_{A2}|$ oznacza liczbę elementów zbioru S^{is}_{A2} spektrum idealnego. Wynika stąd, że błędy negatywne w zbiorze S_{A2} mogą doprowadzić do odrzucenia pewnych elementów ze zbioru S_{A1}. Aby to ograniczyć, sprawdzany jest tylko pierwszy nowy dla ścieżki nukleotyd dodawany przez weryfikowany następnik (przykład z rysunku 6.7). Ponieważ jednak taka sytuacja może się zdażyć, określona jest minimalna liczba wierzchołków do wykorzystania z S_{A1}, tak aby rozwiązanie zostało zaakceptowane.

4) Błędy negatywne oraz pozytywne w spektrum. Jest to najtrudniejszy przypadek, jednak najbardziej realistyczny. Zasady weryfikacji są takie same jak poprzednio - tj. tylko pierwszy nowy nukleotyd w następniku jest sprawdzany przez element ze zbioru S_{A2}. Niemożliwe jest tutaj określenie, ile elementów ze zbioru S_{A1} musi znaleźć się w ścieżkach budowanych przez algorytm. Obecność błędów pozytywnych w nieznanej ilości obok błędów negatywnych uniemożliwia precyzyjne obliczenia. Wartość ta jest określana w przybliżeniu, w rozdziale ósmym zostaną przedstawione wyniki testów dla wartości limitu równej 10%, 15% i 20% przyjętych błędów negatywnych. Przypadki te zostaną także porównane z jakością rozwiązań przy dokładnej znajomości liczby tych błędów. Sama idea limitu jest podobna jak w poprzednim algorytmie - nawet założenie występowania 15% czy 20% błędów negatywnych w całym spektrum jest lepsze z punktu widzenia efektywności stosowanego podejścia, niż rozpatrywanie nieograniczonej ich liczby.

Podsumowując temat błędów oraz ich wpływu na algorytm, określenie liczby błędów negatywnych występujących w spektrum ma zasadnicze znaczenie dla efektywności przeszukiwania przestrzeni rozwiązań. Jeżeli nałożenie nowego następnika z ostatnim wierzchołkiem ścieżki jest mniejsze niż maksymalne, oznacza to, że pomaga on w naprawie efektu wystąpienia pewnego błędu negatywnego, który uniemożliwił nałożenie na równo $max_{overlap} = l_1 - 2$ znakach w tym miejscu. Branie pod uwagę wierzchołków o nałożeniu mniejszym niż $max_{overlap}$, znacznie powiększa przestrzeń rozwiązań w omawianym problemie. Jeżeli możliwe jest precyzyjne określenie liczby błędów negatywnych (tj. w przypadku gdy nie ma błędów pozytywnych), algorytm wiedząc, że "naprawił" już odpowiednią liczbę błędów negatywnych, przestaje brać pod uwagę wierzchołki z nałożeniem mniejszym niż $max_{overlap}$, co poprawia jego dalszą efektywność. Niestety, najbardziej realistyczny przypadek to ten, w którym występują wszystkie rodzaje błędów. Jak już wyjaśniono, niemożliwe jest wtedy precyzyjne określenie liczby błędów

negatywnych. Algorytm dla zachowania efektywności przybliża tę wartość, w domyślnym przypadku zakładając istnienie maksymalnie 20% błędów negatywnych. Oznacza to między innymi akceptację rozwiązań, które mając ustalony rozmiar poszukiwanego DNA wykorzystały przynajmniej 80% liczby elementów spektrum idealnego. Porównanie skuteczności między takim podejściem (oraz także wartością limitu równą 10% i 15% błędów), a podejściem ze znajomością precyzyjnej liczby błędów negatywnych zostanie przedstawione w rozdziale z wynikami testów dla algorytmu.

Jeśli chodzi o pseudokod głównej pętli algorytmu, należy dokładnie opisać jeszcze dwie jego sekcje: powrót do poprzednich wierzchołków w celu zbadania innych rozgałęzień przestrzeni rozwiązań oraz dodanie rekonstrukcji DNA do listy rozwiązań. To ostatnie zagadnienie wiąże się zarówno z jej odtworzeniem z dwóch ścieżek dla nukleotydów nieparzystych i parzystych, jak i z weryfikacją końcową takiego superciągu przez elementy zbioru S_{A2}.

Procedura powrotu do poprzednich wierzchołków wywoływana jest w liniach 9, 11 i 14 głównej pętli algorytmu 3. Jest ona wywoływana w przypadku gdy:

1) nowa rekonstrukcja została utworzona ze ścieżek o odpowiedniej długości;

2) nie powiodła się próba dodania nowego wierzchołka z listy kandydatów na następników.

Niezależnie od tego, czy rozwiązanie zostało zaakceptowane czy nie, wywoływany jest przypadek 1). Drugi przypadek może wystąpić z wielu powodów, które stoją za niemożnością dodania wierzchołka do ścieżki. Dla przykładu, lista kandydatów następników może być pusta, wierzchołki na niej zostały już użyte wcześniej lub jedyne dostępne wierzchołki na liście kandydatów rozszerzają ścieżkę do rozmiaru, który powoduje przekroczenie rozmiaru badanego DNA. Jaki by nie był powód niemożliwości dodania nowego wierzchołka do konstruowanego rozwiązania, opisywana teraz sekcja algorytmu odpowiada za wycofanie się do wierzchołka, z którego jest możliwe rozszerzenie którejkolwiek ścieżki w innym "kierunku" w przestrzeni rozwiązań.

Sama zasada działania tej sekcji algorytmu jest następująca: mając dane o aktualnie zbudowanych ścieżkach, oraz listy kandydatów na następników dla każdego dodanego wierzchołka, algorytm wycofuje się do miejsca, gdzie możliwy jest wybór innego wierzchołka z listy. Jest tylko jeden wyjątek, który jest także tutaj obsługiwany: dodano ostatni wierzchołek rekonstruujący ścieżkę o maksymalnej dopuszczalnej długości, natomiast druga ścieżka osiągnęła już taką długość w poprzednim kroku pętli głównej. W tym wypadku, niezależnie czy rozwiązanie zostało zaakceptowane czy nie, algorytm przed wycofaniem się do poprzedniego wierzchołka sprawdza, czy na ostatniej stworzonej liście kandydatów znajduje się wierzchołek, który można podmienić z aktualnie użytym

tak, aby zachować wymaganą długość ścieżki (czyli musi on mieć ten sam rozmiar nałożenia co wierzchołek, który spowodował wydłużenie ścieżki do maksymalnego rozmiaru). Teoretycznie, w przypadku z błędami, maksymalnie mogą wystąpić 4 różne następniki kończące ścieżkę, w zależności od weryfikacji oraz ich wystąpienia w części S_{A1} spektrum. Jeśli taka podmiana się uda, procedura powrotu kończy pracę, a algorytm wraca do głównej pętli, w której w następnej iteracji nie doda nowego wierzchołka (ponieważ ścieżka wciąż ma maksymalny rozmiar), lecz sprawdzi, czy nowa ścieżka tworzy wraz z drugą nowy łańcuch DNA, który można dodać do listy rozwiązań.

Jeżeli nie można zamienić ostatniego wierzchołka tak, aby pasował (tj. spełniał opisane wcześniej wymagania w stosunku do następnika), algorytm wraca do poprzedniego odwiedzonego wierzchołka ścieżki. W tym kroku zerowane są odpowiednie wartości związane z usuniętym właśnie ze ścieżki wierzchołkiem - np. status jego użycia w ścieżce. Następnie procedura stara się znaleźć inny następnik z wciąż dostępnej listy następników, który można podstawić na miejsce właśnie usuniętego. Jeżeli nie jest to możliwe, następuje przejście do ostatnio dodanego wierzchołka z drugiej ścieżki, ponieważ są one dodawane naprzemiennie. Każdy wierzchołek, z którego listy następników nie można dodać nowego wierzchołka na miejsce usuniętego jest również usuwany ze ścieżki, przy odpowiedniej aktualizacji parametrów rozwiązania. Są to: status użycia wierzchołka w ścieżce oraz aktualny poziom "naprawionych" błędów negatywnych. Całość procedury jest powtarzana tak długo, aż uda się dodać nowy nieużywany dotąd następnik na miejsce ostatnio usuniętego. Jeśli procedura powrotu wykryje, że skończyła się lista kandydatów na następników w pierwszej liście kandydatów, czyli związanej z wierzchołkiem początkowym ścieżki, do pętli głównej algorytmu wysyłana jest informacja o wyczerpaniu przestrzeni rozwiązań.

Rekonstrukcja DNA jest sprawą prostą. Dwie ścieżki rekonstruują odpowiednio nukleotydy nieparzyste i parzyste, wystarczy je więc naprzemiennie odczytać aby odtworzyć sekwencję DNA. "Zgodność" obu ścieżek została zapewniona przez procedurę weryfikacji. Przed dodaniem nowego łańcucha DNA do listy jest ono weryfikowane ponownie. Ta procedura jest zależna od przypadku występowania dwóch głównych typów błędów hybrydyzacji. W przypadku, gdy błędy te nie występują, lub występują tylko błędy negatywne wynikające z powtórzeń, można precyzyjnie określić liczbę elementów spektrum DNA, które muszą zostać wykorzystane. Przy obu rodzajach błędów wartość minimalnej liczby elementów, które należało użyć z S_{A1} jest ustalona w przybliżeniu.

Ponowna weryfikacja spełnia dwie funkcje. Jeżeli nie występują błędy pozytywne, jest ona w stanie sprawdzić rozwiązanie wszystkimi elementami z S_{A2}. Tutaj działa ona w przeciwieństwie do przypadku, w którym weryfikacja w każdym kroku sprowadzała się do sprawdzenia tylko pierwszego nukleotydu rozszerzającego daną ścieżkę. Po drugie, w przypadku gdy występują oba rodzaje błędów i część elementów wciąż nie może być

potwierdzona z uwagi na błędy negatywne, weryfikacja ta dostarcza informację o ilości elementów weryfikacyjnych, które występują w rozwiązaniu (tj. nakładają się na jego fragmenty zgodnie z zasadami nakładania dla ciągów typu Alternating). W przypadku rozwiązań niejednoznacznych stanowi to dodatkową informację umożliwiającą uszeregowanie rozwiązań otrzymanych w procesie przeszukiwania przestrzeni rozwiązań.

Rozdział 7

Binary Chip

7.1 Wstęp

Trzecim chipem do sekwencjonowania przez hybrydyzację, który zostanie w niniejszym rozdziale opisany wraz z zaproponowanym dla niego algorytmem jest chip typu Binary. Sposób jego budowy został pokazany w artykule [40]. Chip ten wykorzystuje inny alfabet znaków do zapisu sond, niż dotychczas stosowany 5-literowy {A, C, G, T, X}. Jest to alfabet 8-literowy {W, S, R, Y, A, C, G, T}, który prościej będzie już teraz podzielić na dwa 6-literowe alfabety: {W, S, A, C, G, T} oraz {R, Y, A, C, G, T}. Dalej będą one nazywane alfabetami typu Binary. Litery W, S, R i Y są w rzeczywistości zbiorami 2-elementowymi, podobnie jak w poprzednich dwóch omawianych chipach znak X był zbiorem 4-elementowym. Szczegóły konstrukcji zaproponowanego w [40] chipu Binary prezentuje następny podrozdział, poświęcony przede wszystkim dokładnej budowie jego sond.

7.2 Konstrukcja i charakterystyka chipu do sekwencjonowania

Zanim zostanie opisana budowa elementów chipu, zostaną teraz opisane wszystkie zbiory alfabetów niezbędne do zrozumienia idei stojącej za konstrukcją chipu typu Binary.

Chip składa się z dwóch części, z których w każdej elementy zbudowane są na bazie alfabetów 6-literowych:

{W, S, A , C , G , T } oraz {R, Y, A , C , G , T }

Jak widać, podzbiór $N = \{A, C, G, T\}$ jest wspólnym alfabetem opisującym "normalne" nukleotydy w obu alfabetach 6-znakowych. Symbole W, S, R i Y to w rzeczywistości zbiory 2-elementowe, zawierające różne pary liter alfabetu N. Poniżej przedstawiono wszystkie zbiory:

W = {A , T } ; zbiór ten reprezentuje nukleotydy słabe

S = {C , G } ; zbiór ten reprezentuje nukleotydy silne

R = {A , G } ; zbiór ten reprezentuje puryny

Y = {C , T } ; zbiór ten reprezentuje pirymidyny

Teraz można już opisać sam chip. Składa się on z dwóch części, których elementy są kodowane następująco:

$$\underbrace{\{W,S\},\{W,S\},...,\{W,S\}}_{k},N \text{ oraz } \underbrace{\{R,Y\},\{R,Y\},...,\{R,Y\}}_{k},N$$

Pojemność każdej połowy chipu, przy założeniu występowania wszystkich kombinacji liter alfabetów 6-literowych dla elementów o długościach:

$$l_{bin} = k + 1 \tag{7.1}$$

to $2^k \cdot 4$, tak więc pojemność całego chipu typu Binary dana jest wzorem:

$$|C_{bin}(k)| = 2 \cdot 2^k \cdot 4 \tag{7.2}$$

Dla danych wartości parametrów n i k, określających odpowiednio długość badanego DNA oraz długość elementów chipu (oraz jego pojemność) spektrum idealne bez powtórzeń posiada dwa podzbiory o pojemnościach:

$$|S^{is}_{B1}| = n - (k+1) + 1 = n - k \text{ oraz } |S^{is}_{B2}| = n - (k+1) + 1 = n - k \tag{7.3}$$

Rozpatrywany dalej algorytm zakłada, że spektrum pochodzi z eksperymentu hybrydyzacyjnego z udziałem chipu typu Binary posiadającego wszystkie sondy dla wzorów o zadaj długości l_{bin}. Na przykład chip dla którego $k = 2$ będzie posiadać $|C_{bin}(k)| = 2 \cdot 2^k \cdot 4 = 8 \cdot 2^2 = 32$ sond. Połowy chipu składać się będą z 16 sond. Pierwsza z nich zawierać będzie sondy:

$$W\ W\ A\ ;\ W\ S\ A\ ;\ S\ W\ A\ ;\ S\ S\ A$$
$$W\ W\ C\ ;\ W\ S\ C\ ;\ S\ W\ C\ ;\ S\ S\ C$$
$$W\ W\ G\ ;\ W\ S\ G\ ;\ S\ W\ G\ ;\ S\ S\ G$$
$$W\ W\ T\ ;\ W\ S\ T\ ;\ S\ W\ T\ ;\ S\ S\ T$$

Zbiór drugiej połowy sond posiada następujące elementy:

$$R\ R\ A\ ;\ R\ Y\ A\ ;\ Y\ R\ A\ ;\ Y\ Y\ A$$
$$R\ R\ C\ ;\ R\ Y\ C\ ;\ Y\ R\ C\ ;\ Y\ Y\ C$$
$$R\ R\ G\ ;\ R\ Y\ G\ ;\ Y\ R\ G\ ;\ Y\ Y\ G$$
$$R\ R\ T\ ;\ R\ Y\ T\ ;\ Y\ R\ T\ ;\ Y\ Y\ T$$

Liczba oligonukleotydów w każdej nieklasycznej sondzie chipu jest również zależna od parametru k:

$$n_{oligo} = 2^k \tag{7.4}$$

Dla przykładu, dla $k = 4$ w każdej sondzie znaleźć się może 16 oligonukleotydów zbudowanych nad naturalnym alfabetem {A, C, G, T}, jeżeli podejście nie będzie wykorzystywać nukleotydów zdegenerowanych opisanych w rozdziale 4 rozprawy. Hipotetyczna sonda opisana wzorem $WSWSA$ musi posiadać wtedy następujące oligonukleotydy:

$$A\ C\ A\ C\ A\ ;\ A\ C\ A\ G\ A\ ;\ A\ C\ T\ C\ A\ ;\ A\ C\ T\ G\ A$$
$$A\ G\ A\ C\ A\ ;\ A\ G\ A\ G\ A\ ;\ A\ G\ T\ C\ A\ ;\ A\ G\ T\ G\ A$$
$$T\ C\ A\ C\ A\ ;\ T\ C\ A\ G\ A\ ;\ T\ C\ T\ C\ A\ ;\ T\ C\ T\ G\ A$$
$$T\ G\ A\ C\ A\ ;\ T\ G\ A\ G\ A\ ;\ T\ G\ T\ C\ A\ ;\ T\ G\ T\ G\ A$$

Należy nadmienić, że tak jak poprzednio, realizacja budowy chipu nie jest ważna dla zaproponowanego algorytmu. Sposób powyższy został tu przykładowo opisany celem wyjaśnienia podstawowych charakterystyk chipu i spektrum. Jeżeli możliwe byłoby zbudowanie chipu przy użyciu nukleotydów zdegenerowanych, spektrum dane na wejściu algorytmu wciąż zawierałoby identyczną informację o sondach, tj. wzór każdej sondy, która hybrydyzowała.

W opisywanych chipie nie ma dedykowanych elementów weryfikacyjnych, jak miało to miejsce dla chipów typu Gapped i Alternating. Z obu części spektrum powstają dwa grafy, w których mechanizm weryfikacji rozwiązany jest inaczej niż poprzednio, a jego dokładny opis znajduje się w części rozdziału poświęconej opisowi algorytmu.

Należy opisać jeszcze jedną cechę sond chipu, mianowicie sposób uzyskania elementu

zbudowanego nad alfabetem 4-literowym {A, C, G, T}, z dwóch elementów stworzonych na bazie różnych alfabetów typu Binary. Zbiory W oraz S mają tylko po jednym elemencie wspólnym ze zbiorami R i Y. Elementy te zostały rozpisane poniżej:

$$W \cup R = A$$

$$W \cup Y = T$$

$$S \cup R = G$$

$$S \cup Y = C$$

Dla przykładu, mając dwa elementy: $W\ S\ W\ W\ S\ A$ oraz $R\ R\ Y\ Y\ Y\ A$, można nałożyć je całkowicie na siebie, otrzymując fragment DNA opisany ciągiem znaków: $A\ G\ T\ T\ C\ A$

7.3 Zagadnienia prawdopodobieństwa rozgałęzień

W niniejszym podrozdziale przedstawione zostaną teoretyczne obliczenia związane z charakterystyką chipu typu Binary, pochodzące z pracy [40].

Poszukujemy prawdopodobieństwa określanego dalej jako $p(C_{bin}(k), n)$. Jak poprzednio, rozważmy sekwencję F, dla której w momencie dodania nukleotydu na pozycji m pojawia się więcej niż jedno możliwe rozszerzenie. Są to odpowiednio: nukleotyd N_m, oraz pewien inny nukleotyd Y. Załóżmy, że $m \geqslant k$ oraz $k \ll n \ll |C_{bin}(k)|$. Niech $F = N_1\ N_2\ ...\ N_{m-1}$, $V = N_{m-k+2}...N_{m-1}$, a ciąg VY będzie rozszerzeniem sekwencji V, innym niż VN_m. Prawdopodobieństwo znalezienie każdego takiego elementu w chipie $C_{bin}(k)$ to:

$$\frac{1}{4 \cdot 2^k} \tag{7.5}$$

Kolejne prawdopodobieństwo (przybliżone) określa sytuację, w której spektrum sekwencji F o długości n nie posiada elementu VY ($l_bin = k + 1$):

$$(1 - \frac{1}{4 \cdot 2^k})^{n-k} \tag{7.6}$$

Ponieważ w przypadku omawianego chipu mamy do czynienia z dwoma elementami o równej długości, zbudowanych na dwóch różnych, 6-literowych alfabetach, lecz opisujących jeden podłańcuch DNA, prawdopodobieństwo, że spektrum F zawiera obydwa to:

$$(1-(1-\frac{1}{4\cdot 2^k})^{n-k})\cdot(1-(1-\frac{1}{4\cdot 2^k})^{n-k})\approx\frac{n^2}{4\cdot 2^k\cdot 4\cdot 2^k} \tag{7.7}$$

Zakładając, że $p(C_{bin}(k),n)=P\{V'Y_i\in S(C_{bin}(k),F)$ oraz $V''Y_i\in S(C_{bin}(k),F)\}$, gdzie V' oznacza sekwencję nad alfabetem {W, S, A, C, G, T}, natomiast V'' sekwencję nad alfabetem {R, Y, A, C, G, T} otrzymujemy:

$$p(C_{bin}(k),n)\approx 1-(1-\frac{n^2}{4\cdot 2^k\cdot 4\cdot 2^k})^3\approx 3\cdot\frac{n}{4\cdot 2^k}\cdot\frac{n}{4\cdot 2^k}=\frac{12n^2}{|C_{bin}(k)|^2} \tag{7.8}$$

Stąd:

$$n_{max}(C_{bin}(k),p)\approx\frac{1}{\sqrt{12}}\cdot|C_{bin}(k)|\sqrt{p} \tag{7.9}$$

7.4 Problemy oraz ich złożoność obliczeniowa

7.4.1 Podstawowe definicje

Def. 7.4.1. Ciągi typu Binary
Zbiór S_{B1} zawiera wszystkie elementy zbudowane nad alfabetem {W, S, A, C, G, T} o długości $l=k+1$.
Zbiór S_{B2} zawiera wszystkie elementy zbudowane nad alfabetem {R, Y, A, C, G, T} o długości $l=k+1$. Elementy obu zbiorów spełniają warunek, w myśl którego litera należąca do wspólnego podalfabetu {A, C, G, T} i tylko ona, znajduje się na ostatniej pozycji każdego elementu
Ciągi typu Binary są elementami zbioru $S_1\cup S_2$

Def. 7.4.2. Nakładanie ciągów typu Binary
Ciągi p o długości l_p oraz q o długości l_q zbudowane nad alfabetami {W, S, A, C, G, T} lub {R, Y, A, C, G, T} nakładają się na pewnej długości $r\leqslant min(l_p,l_q)$, jeżeli dla dwóch ich podciągów o długości r, p_{a+i} oraz q_{b+i} dla każdego $i=0,1,...,r-1$ spełniony jest któryś z warunków:

1) jeżeli $p_{a+i}=W$, to $q_{b+i}\in$ {A, T, W, R, Y}

2) jeżeli $p_{a+i}=S$, to $q_{b+i}\in$ {C, G, S, R, Y}

3) jeżeli $p_{a+i}=R$, to $q_{b+i}\in$ {A, G, R, W, S}

4) jeżeli $p_{a+i} = Y$, to $q_{b+i} \in \{\text{C, T, Y, W, S}\}$

5) jeżeli $p_{a+i} = A$, to $q_{b+i} \in \{\text{A, W, R}\}$

6) jeżeli $p_{a+i} = C$, to $q_{b+i} \in \{\text{C, S, Y}\}$

7) jeżeli $p_{a+i} = G$, to $q_{b+i} \in \{\text{G, S, R}\}$

8) jeżeli $p_{a+i} = T$, to $q_{b+i} \in \{\text{T, W, Y}\}$

Zmienne a i b to indeksy pierwszego znaku podciągów w ciągach, odpowiednio p i q. Jeden z indeksów (a lub b) musi być równy 1, co zapewnia, że jeden z podciągów o których tutaj mowa stanowi początek ciągu p lub q, od którego pochodzi. Efektem nałożenia się p i q jest superciąg t ciągów p i q, w którym każdy znak $t_{a+b+i-1}$ (tj. znak części wspólnej p i q w t) stworzony jest następująco:

1) jeżeli $p_{a+i} = q_{b+i}$, wtedy $t_{a+b+i-1} = p_{a+i} = q_{b+i}$

2) jeżeli $p_{a+i} = W$ i $q_{b+i} = R$ (lub odwrotnie), wtedy $t_{a+b+i-1} = A$

3) jeżeli $p_{a+i} = W$ i $q_{b+i} = Y$ (lub odwrotnie), wtedy $t_{a+b+i-1} = T$

4) jeżeli $p_{a+i} = S$ i $q_{b+i} = R$ (lub odwrotnie), wtedy $t_{a+b+i-1} = G$

5) jeżeli $p_{a+i} = S$ i $q_{b+i} = Y$ (lub odwrotnie), wtedy $t_{a+b+i-1} = C$

6) jeżeli $p_{b+i} = W$ oraz $q_{a+i} \in \{A, T\}$ (lub odwrotnie), wtedy $t_{a+b+i-1} = q_{a+i}$

7) jeżeli $p_{b+i} = S$ oraz $q_{a+i} \in \{C, G\}$ (lub odwrotnie), wtedy $t_{a+b+i-1} = q_{a+i}$

8) jeżeli $p_{b+i} = R$ oraz $q_{a+i} \in \{A, G\}$ (lub odwrotnie), wtedy $t_{a+b+i-1} = q_{a+i}$

6) jeżeli $p_{b+i} = Y$ oraz $q_{a+i} \in \{C, T\}$ (lub odwrotnie), wtedy $t_{a+b+i-1} = q_{a+i}$

Jeśli na przykład nakładamy na siebie ciągi $SWWWWA$ oraz $WWWWWG$ (podciąg $WWWWA$ to postfix w $SWWWWA$, który nakłada się z $WWWWW$ czyli prefixem z $WWWWWG$), ciąg wynikowy ma postać: $SWWWWAG$.

Def. 7.4.3. Uogólniony superciąg typu Binary
Dla danego zbioru S_B ciągów typu Binary, uogólnionym superciągiem o długości n typu Binary jest taki superciąg, który składa się ze wszystkich elementów zbioru S_B nakładających się na siebie zgodnie z definicją nakładania ciągów typu Binary.

7.4.2 Problem Binary SBH bez błędów hybrydyzacji

Def. 7.4.4. Idealne spektrum problemu Binary SBH (BSBH)
Idealne spektrum składa się ze wszystkich ciągów typu Binary o długościach $l = k + 1$ sekwencji DNA o długości n. W takim wypadku moce zbiorów $S_{B1}^{(is)}$ oraz $S_{B2}^{(is)}$ spełniają nierówność: $|S_{B1}^{(is)}| \leqslant n - l + 1$, $|S_{B2}^{(is)}| \leqslant n - l + 1$.
W spektrum idealnym każdy element nakłada się na podciąg o długości l sekwencji DNA zgodnie z definicją nakładania się ciągów typu Binary. Na przykład element $WSWSA$ nakłada się na podciąg $ACAGA$ sekwencji DNA.

Problem 7.4.1. Problem BSBH bez błędów w wersji decyzyjnej (BSBH-efd, error-free, decision)
Instancja: zbiór $S_B = S_{B1} \cup S_{B2}$ taki, że $S_{B1} = S_{B1}^{(is)}$ oraz $S_{B2} = S_{B2}^{(is)}$, będący idealnym spektrum BSBH, długość n sekwencji DNA, $|S_{B1}| = n - l + 1$, $|S_{B2}| = n - l + 1$.
Odpowiedź: TAK, jeżeli istnieje uogólniony superciąg typu Binary o długości n zbudowany nad alfabetem {A, C, G, T}, utworzony ze wszystkich elementów zbiorów S_{B1} i S_{B2}.

W przypadku idealnego spektrum BSBH problem jest łatwy obliczeniowo. Musi istnieć przynajmniej jeden taki superciąg, tak więc zbiór wszystkich instancji problemu BSBH jest równy zbiorowi instancji tego problemu, dla których odpowiedź brzmi TAK, $D_{\Pi_{BSBH-efd}} = Y_{\Pi_{BSBH-efd}}$.

Problem 7.4.2. Problem BSBH bez błędów w wersji przeszukiwania (BSBH-efs, error-free, search)
Instancja: zbiór $S_B = S_{B1} \cup S_{B2}$ taki, że $S_{B1} = S_{B1}^{(is)}$ oraz $S_{B2} = S_{B2}^{(is)}$, będący idealnym spektrum BSBH, długość n sekwencji DNA, $|S_{B1}| = n - l + 1$, $|S_{B2}| = n - l + 1$.
Odpowiedź: uogólniony superciąg typu Binary o długości n zbudowany nad alfabetem {A, C, G, T}, utworzony ze wszystkich elementów zbiorów S_{B1} i S_{B2}.

7.4.3 Binary SBH z błędami negatywnymi

Def. 7.4.5. Idealne multispektrum problemu Binary SBH
Idealne multispektrum składa się ze wszystkich ciągów typu Binary o długościach $l = k+$ 1 z możliwością powtórzeń. W przypadku multispektrum idealnego moce multizbiorów $S_{B1}^{(im)}$ oraz $S_{B2}^{(im)}$ są równe $|S_{B1}^{(im)}| = |S_{B2}^{(im)}| = n - l + 1$ gdzie n jest długością DNA.
Każdy ciąg e ze zbioru $S = S_{B1}^{(im)} \cup S_{B2}^{(im)}$ powtarza się w spektrum tyle razy, ile razy występuje w sekwencji DNA jej podciąg o długości l, na który ciąg e nakłada się zgodnie z definicją nakładania dla ciągów typu Binary.

Problem 7.4.3. Problem BSBH z błędami negatywnymi w wersji decyzyjnej (BSBH-ned, negative errors, decision)
Instancja: zbiór $S_B = S_{B1} \cup S_{B2}$ taki, że $S_{B1} \subset S_{B1}^{(im)}$ oraz $S_{B2} \subset S_{B2}^{(im)}$, będący podzbiorem bez powtórzeń idealnego multispektrum BSBH, długość n sekwencji DNA.
Odpowiedź: TAK, jeżeli istnieje uogólniony superciąg typu Binary zbudowany nad alfabetem {A, C, G, T} o długości mniejszej lub równej n, utworzony ze wszystkich elementów zbiorów S_{B1} i S_{B2}.

Zbiór wszystkich instancji powyższego problemu *nie* jest równy zbiorowi instancji, dla których odpowiedź brzmi TAK. Zakładając warunek, że uogólniony superciąg musi być zbudowany nad alfabetem {A, C, G, T}, mogą istnieć instancje będące pozbiorem idealnego multispektrum, dla których niemożliwe jest zrekonstruowanie superciągu składającego się z liter z alfabetu {A, C, G, T} (np. znaczna różnica w liczbie elementów jednego z dwóch podzbiorów spektrum BSBH). Dlatego też zbiór wszystkich instancji tego problemu składa się z instancji, dla których odpowiedź brzmi TAK oraz instancji, dla których odpowiedź brzmi NIE: $D_{\Pi_{BSBH-ned}} = Y_{\Pi_{BSBH-ned}} \cup N_{\Pi_{BSBH-ned}}$.

Problem 7.4.4. Problem BSBH z błędami negatywnymi w wersji przeszukiwania (BSBH-nes, negative errors, search)
Instancja: zbiór $S_B = S_{B1} \cup S_{B2}$ taki, że $S_{B1} \subset S_{B1}^{(im)}$ oraz $S_{B2} \subset S_{B2}^{(im)}$, będący podzbiorem bez powtórzeń idealnego multispektrum BSBH, długość n sekwencji DNA.
Odpowiedź: uogólniony superciąg typu Binary zbudowany nad alfabetem {A, C, G, T} o długości mniejszej lub równej n, utworzony ze wszystkich elementów zbiorów S_{B1} i S_{B2}.

Zbiór wszystkich instancji problemu przeszukiwania jest równy zbiorowi wszystkich instancji problemu decyzyjnego: $D_{\Pi_{BSBH-nes}} = D_{\Pi_{BSBH-ned}}$.

7.4.4 Binary SBH z błędami pozytywnymi

Definicje problemów sformułowane poniżej wykorzystują definicję idealnego spektrum BSBH 7.4.4 oraz idealnego multispektrum dla BSBH 7.4.5.

Problem 7.4.5. Problem BSBH z błędami pozytywnymi w wersji decyzyjnej (BSBH-ped, positive errors, decision)
Instancja: zbiór $S_B = S_{B1} \cup S_{B2}$ taki, że $S_{B1}^{(is)} = S_{B1}^{(im)} \subset S_{B1}$ oraz $S_{B2}^{(is)} = S_{B2}^{(im)} \subset S_{B2}$, zawierający jako podzbiór spektrum idealne BSBH, długość n sekwencji DNA.
Odpowiedź: TAK, jeżeli istnieje uogólniony superciąg typu Binary o długości n zbudowany nad alfabetem {A, C, G, T}, utworzony z $|S_{B1}^{(is)}|$ elementów podzbioru S_{B1} oraz $|S_{B1}^{(is)}|$ elementów podzbioru S_{B2}.

Zbiór wszystkich instancji powyższego problemu jest równy zbiorowi instancji, dla których odpowiedź brzmi TAK. Idealne spektrum problemu BSBH jest podzbiorem spektrum problemu BSBH-ned. Wiadomo, że z idealnego spektrum BSBH można otrzymać uogólniony superciąg o długości n, tak więc złożoność powyższego problemu decyzyjnego jest wielomianowa: odpowiedź zawsze brzmi TAK. Zbiór wszystkich instancji powyższego problemu jest równy zbiorowi instancji, dla których odpowiedź brzmi TAK: $D_{\Pi_{BSHB-ped}} = Y_{\Pi_{BSBH-ped}}$.

Problem 7.4.6. Problem BSBH z błędami pozytywnymi w wersji przeszukiwania, (BSBH-pes, positive errors, search)
Instancja: zbiór $S_B = S_{B1} \cup S_{B2}$ taki, że $S_{B1}^{(is)} = S_{B1}^{(im)} \subset S_{B1}$ oraz $S_{B2}^{(is)} = S_{B2}^{(im)} \subset S_{B2}$, zawierający jako podzbiór spektrum idealne BSBH, długość n sekwencji DNA.
Odpowiedź: uogólniony superciąg typu Binary o długości n zbudowany nad alfabetem {A, C, G, T}, utworzony z $|S_{B1}^{(is)}|$ elementów podzbioru S_{B1} oraz $|S_{B1}^{(is)}|$ elementów podzbioru S_{B2}.

Zbiór wszystkich instancji tego problemu jest równy zbiorowi wszystkich instancji problemu decyzyjnego: $D_{\Pi_{BSHB-pes}} = D_{\Pi_{BSBH-ped}}$. Innymi słowy jest to zbiór wszystkich instancji z odpowiedzią TAK.

Problem 7.4.7. Problem quasi-pozytywnego BSBH w wersji decyzyjnej (qBSBH-ped, positive errors, decision)
Instancja: zbiór $S_B = S_{B1} \cup S_{B2}$, zawierający ciągi typu Binary o długości l, długość n badanej sekwencji DNA.
Odpowiedź: TAK, jeżeli istnieje uogólniony superciąg typu Binary o długości n zbudowany nad alfabetem {A, C, G, T}, utworzony z $n - l + 1$ elementów zbioru S_{B1} oraz $n - l + 1$ elementów zbioru S_{B2}.

Zbiór wszystkich instancji z odpowiedzią TAK problemu qBSBH-ped jest równy zbiorowi wszystkich instancji z odpowiedzią TAK dla problemu BSBH-ped. Zbiór wszystkich instancji BSBH-ped jest jednak podzbiorem właściwym zbioru wszystkich instancji problemu qBSBH-ped, ponieważ ten drugi zawiera także instancje, dla których odpowiedź brzmi NIE, $Y_{\Pi_{qBSBH-ped}} = Y_{\Pi_{BSBH-ped}}$, $D_{\Pi_{BSBH-ped}} \subset D_{\Pi_{qBSBH-ped}}$

Twierdzenie 7.4.1. Problem qBSBH-ped jest silnie NP-zupełny.

Dowód Instancja problemu qBSBH-pes z błędami pozytywnymi zostanie transformowana z instancji problemu skierowanej ścieżki Hamiltona pomiędzy dwoma wierzchołkami (DHPBTV), podobnie jak w dowodzie [10] dla problemu klasycznego sekwencjonowania przez hybrydyzację z błędami pozytywnymi. Transformacja z artykułu zostanie tutaj przytoczona w całości, ponieważ jest punktem wyjścia do dalszej transformacji

mającej na celu otrzymanie instancji problemu qBSBH-ped. Transformacja oryginalna z [10] jest w nieznacznym stopniu zmodyfikowana, a zmiana dotyczy ostatnich dwóch elementów, które budują superciąg będący odpowiednikiem ścieżki Hamiltona w grafie. W artykule superciąg taki musi kończy się fragmentem $...TTTTG$, natomiast po modyfikacji superciąg kończyć się będzie fragmentem $...TTTGG$. Transformacja z artykułu po wspomnianej modyfikacji przedstawia się następująco:

Problem 7.4.8. Skierowana ścieżka Hamiltona między dwoma wierzchołkami, DHPBTW
Instancja: graf skierowany $H = (V, A)$, z zaznaczonym wierzchołkiem początkowym i końcowym s i t.
Odpowiedź: TAK, jeżeli istnieje ścieżka Hamiltona pomiędzy s i t.

Mając daną instancję DHPBTV postępujemy następująco:

1) każdemu wierzchołkowi $v \in V$ przydzielamy unikalną nazwę $e(v)$ o długości $\lceil log_2|V| \rceil$ nad alfabetem {A, C};

2) definiujemy $l = 2\lceil log_2|V| \rceil + 2$ jako długość wszystkich elementów w S;

3) budujemy zbiór S tak, że dla każdego wierzchołka $v \in V$ budujemy jeden oligonukleotyd w formie $e(v) \cdot G \cdot e(v) \cdot T$ ($\cdot$ oznacza konkatenację, G oraz T to nukleotydy);

4) do S dodajemy $l-1$ oligonukleotydów dla każdego łuku $(u, v) \in X$, oligonukleotydy w formie $u_2 \cdot u_3 \cdot \cdots \cdot v_1, u_3 \cdot u_4 \cdot \cdots \cdot v_1 \cdot v_2, \ldots, u_l \cdot v_1 \cdot \cdots \cdot v_{l-2} \cdot v_{l-1}$ (gdzie $u_1 \cdot u_2 \cdot \cdots \cdot u_l$ to oligonukleotyd oznaczający wierzchołek u);

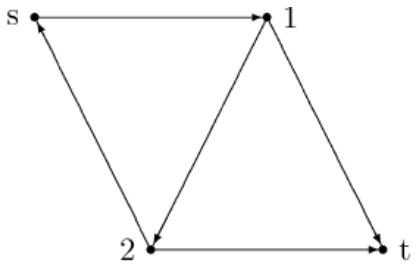
5) do S dodajemy l oligonukleotydów startowych w formie $g_1 \cdot g_2 \cdot \cdots \cdot g_{l-1}, g_2 \cdot g_3 \cdot \cdots \cdot g_l \cdot s_1, \ldots, g_l \cdot s_1 \cdot \cdots \cdot s_{l-2} \cdot s_{l-1}$ (gdzie $g_i = G, 1 \leqslant i \leqslant l$ oraz $s_1 \cdot s_2 \cdot \cdots \cdot s_l$ to oligonukleotyd reprezentujący wierzchołek startowy s);

6) do S dodajemy l oligonukleotydów końcowych w formie $t_2 \cdot t_3 \cdot \cdots \cdot t_l \cdot k_1, t_3 \cdot t_4 \cdot \cdots \cdot k_1 \cdot k_2, \ldots, t_l \cdot k_1 \cdot \cdots \cdot k_{l-2} \cdot k_{l-1}, k_1 \cdot \cdots \cdot k_{l-2} \cdot k_{l-1} \cdot G, k_2 \cdot \cdots \cdot k_{l-2} \cdot G \cdot G$ (gdzie $k_i = T, 1 \leqslant i \leqslant l-1$ oraz $t_1 \cdot t_2 \cdot \cdots \cdot t_l$ to oligonukleotyd reprezentujący wierzchołek końcowy t).

Jak poprzednio, rysunek 7.1 z pracy [10] posłuży jako przykład grafu, dla którego wykonywane będą kolejne kroki transformacji:

Efektem transformacji jest następujący zbiór S ciągów:

1) Odpowiedniki wierzchołków (kolejno s 1 2 t):
$AAGAAT\ ACGACT\ CAGCAT\ CCGCCT$



RYSUNEK 7.1: Graf z instancji problemu DHPBTW [10]

2) Elementy prowadzące do wierzchołka s (startowa ścieżka łukowa):
$GGGGGG\ GGGGGA\ GGGGAA\ GGGAAG\ GGAAGA\ GAAGAA$

3) Elementy wychodzące z wierzchołka t (końcowa ścieżka łukowa):
$CGCCTT\ GCCTTT\ CCTTTT\ CTTTTT\ TTTTTG\ TTTTGG$

4) Elementy tworzące łuki grafu (ścieżki łukowe):
(s,1) - $AGAATA$, $GAATAC$, $AATACG$, $ATACGA$, $TACGAC$
(1,2) - $CGACTC$, $GACTCA$, $ACTCAG$, $CTCAGC$, $TCAGCA$
(1,t) - $CGACTC$, $GACTCC$, $ACTCCG$, $CTCCGC$, $TCCGCC$
(2,s) - $AGCATA$, $GCATAA$, $CATAAG$, $ATAAGA$, $TAAGAA$
(2,t) - $AGCATC$, $GCATCC$, $CATCCG$, $ATCCGC$, $TCCGCC$

Mając dany zbiór S transformowany będzie on do zbiorów będących instancją problemu pozytywnego-qBSBH. Jej elementy przy maksymalnym nałożeniu na siebie, będą mogły utworzyć uogólniony superciąg, będący rozwiązaniem dla problemu DHPBTW. Wszystkie elementy zbioru S zostają zamienione na dwa nowe ciągi znaków każdy, według schematu:

1) Dla każdej litery na pozycji $i = 1, 2, ..., l-1$ w elemencie $e \in S$, gdzie l jest jego długością, litery w każdym elemencie e są odwzorowywane w dwóch nowych elementach według schematu:

   litera A zmieniana jest na WW i RR

   litera C zmieniana jest na WS i RY

   litera G zmieniana jest na SS i YR

   litera T zmieniana jest na SW i YY

2) litera na pozycji $i = l$ w elemencie e jest zamieniana według schematu:

   litera A zmieniana jest na WA i RA

   litera C zmieniana jest na WC i RC

litera G zmieniana jest na SG i YG

litera T zmieniana jest na ST i YT

W efekcie powstanie zbiór Z elementów o następującej strukturze (pre-S oznacza elementy początkowe, post-T elementy końcowe, reszta oznaczeń tak jak w poprzednim przykładzie):

```
Odpowiedniki wierzchołków:
s:WWWWSSWWWWST 1:WWWSSSWWWSST 2:WSWWSSWSWWST t:WSWSSSWSWSST
  RRRRYRRRRRYT   RRRYYRRRRYYT   RYRRYRRYRRYT   RYRYYRRYRYYT

Ścieżki łukowe:
pre-S: SSSSSSSSSSSG, SSSSSSSSSSWA, SSSSSSSSWWWA
       YRYRYRYRYRYG, YRYRYRYRYRRA, YRYRYRYRRRRA,
       SSSSSSWWWWSG, SSSSWWWWSSWA, SSWWWWSSWWWA
       YRYRYRRRRRYG, YRYRRRRRYRRA, YRRRRRYRRRRA

post-t:WSSSWSWSSWST, SSWSWSSWSWST, WSWSSWSWSWST,
       RYYRRYRYYYYT, YRRYRYYYYYYT, RYRYYYYYYYYT,
       WSSWSWSWSWST, SWSWSWSWSWSG, SWSWSWSWSSSG
       RYYYYYYYYYYT, YYYYYYYYYYYG, YYYYYYYYYRYG

(s,1): WWSSWWWWSWWA, SSWWWWSWWWWC, WWWWSWWWWSSG,
       RRYRRRRRYYRA, YRRRRRYYRRRC, RRRRYYRRRYYG,
       WWSWWWWSSSWA, SWWWWSSSWWWC
       RRYYRRRYYRRA, YYRRRYYRRRRC

(1,2): WSSSWWWSSWWC, SSWWWSSWWSWA, WWWSSWWSWWSG,
       RYYRRRRYYYRC, YRRRRYYYRYRA, RRRYYYRYRRYG,
       WSSWWSWWSSWC, SWWSWWSSWSWA
       RYYYRYRRYRRC, YYRYRRYRRYRA

(1,t): WSSSWWWSSWWC, SSWWWSSWWSWC, WWWSSWWSWSSG,
       RYYRRRRYYYRC, YRRRRYYYRYRC, RRRYYYRYRYYG
       WSSWWSWSSSWC, SWWSWSSSWSWC
       RYYYRYRYYYRRC, YYRYRYYRRYRC

(2,s): WWSSWSWWSWWA, SSWSWWSWWWWA, WSWWSWWWWWSG,
       RRYRRYRRYYRA, YRRYRRYYRRRA, RYRRYYRRRRYG,
```

```
       WWSWWWWWSSWA, SWWWWWSSWWWA
       RRYYRRRRYRRA, YYRRRRYRRRRA

(2,t): WWSSWSWWSWWC, SSWSWWSWWSWC, WSWWSWWSWSSG,
       RRYRRYRRYYRC, YRRYRRYYRYRC, RYRRYYRYRYYG,
       WWSWWSWSSSSWC, SWWSWSSSSWSWC
       RRYYRYRYYYRRC, YYRYRYYRRYRC
```

Zbiór Z posiada wszystkie niezbędne elementy, aby zbudować superciąg o długości $2n$, jednak tylko z nałożeniem zawsze na $l-2$ znakach. Oznacza to, że jest to wciąż zbiór także z błędami negatywnymi z punktu widzenia qBSBH. Należy więc rozszerzyć go o elementy umożliwiające nałożenia $l-1$ pomiędzy elementami na ścieżkach łukowych. Na wszystkich ścieżkach łukowych należy utworzyć "łączniki" tj. elementy łączące każdą parę elementów na takiej ścieżce, z nałożeniem na $l-1$ znakach. Pierwsze i ostatnie elementy każdej ścieżki łukowej potrzebują łączników do odpowiedników wierzchołków grafu, np. na ścieżce łukowej (s,1) jej pierwszy element poza łącznikiem z elementem drugim, potrzebuje połączenia z odpowiednikiem wierzchołka s. Ostatni element ścieżki łukowej (s,1) poza połączeniem z elementem przedostatnim, tworzy także łącznik z odpowiednikiem wierzchołka 1. Zasada ta nie dotyczy tylko pierwszego elementu na ścieżce $pre-S$ oraz ostatniego na ścieżce $post-t$. Ostatni elementy $pre-S$ tworzy łącznik z odpowiednikiem wierzchołka s, pierwszy element $post-t$ tworzy łącznik z odpowiednikiem wierzchołka t (oraz oczywiście następnym, tj drugim elementem na $post-t$).

Zasada tworzenia tych elementów jest stosunkowo prosta. Zostanie ona też zobrazowana na przykładzie dalej w podrozdziale. Dla dwóch kolejnych elementów na ścieżce łukowej, oraz ostatniego/pierwszego jej elementu z odpowiednikiem wierzchołka, do której ta ścieżka prowadzi/wychodzi należy utworzyć element łączący. Dwa ciągi mające go utworzyć nakładają się na $l-2$ znakach. Efektem nałożenia jest ciąg w o długości $r=l+2$ znaków. Łącznikiem, umożliwiającym nałożenie trzem elementom (czyli razem z nim samym) na $l-1$ znakach jest podciąg z w, zawierający wszystkie jego znaki poza pierwszym i ostatnim. Ostatni znak takiego łącznika jest zamieniany na odpowiednią literę z {A, C, G, T}. Łączniki te są tworzone tak, aby zapewnić nałożenia zgodnie ze zdefiniowaną regułą nakładania. Po zastosowaniu wspomnianych przekształceń otrzymujemy nowy, powiększony zbiór elementów Z (przykład przekształceń podany zostanie pod zbiorem Z):

```
Odpowiedniki wierzchołków:
s:WWWWSSWWWWST 1:WWWSSSWWWSST 2:WSWWSSWSWWST t:WSWSSSWSWSST
  RRRRYRRRRRYT   RRRYYRRRRYYT   RYRRYRRYRRYT   RYRYYRRYRYYT
```

pre-S:
SSSSSSSSSSG,SSSSSSSSSSA,SSSSSSSSSWA,SSSSSSSSWWA,SSSSSSSWWWA,
YRYRYRYRYRYG,RYRYRYRYRYRA,YRYRYRYRYRRA,RYRYRYRYRRRA,YRYRYRYRRRRA,
SSSSSSSWWWWC,SSSSSSWWWWSG,SSSSSWWWWSSA,SSSSWWWWSSWA,SSSWWWWSSWWA,
RYRYRYRRRRRC,YRYRYRRRRRYG,RYRYRRRRRYRA,YRYRRRRRYRRA,RYRRRRRYRRRA,
SSWWWWSSWWWA,SWWWWSSWWWWC
YRRRRRYRRRRA,RRRRRYRRRRRC
post-t:
SWSSSWSWSSWC,WSSSWSWSSWST,SSSWSWSSWSWC,SSWSWSSWSWST,SWSWSSWSWSWC,
YRYYRRYRYYYC,RYYRRYRYYYYT,YYRRYRYYYYYC,YRRYRYYYYYYT,RRYRYYYYYYYC,
WSWSSWSWSWST,SWSSWSWSWSWC,WSSWSWSWSWST,SSWSWSWSWSWC,SWSWSWSWSWSG,
RYRYYYYYYYYT,YRYYYYYYYYYC,RYYYYYYYYYYT,YYYYYYYYYYYC,YYYYYYYYYYYG,
WSWSWSWSWSSC,SWSWSWSWSSSG
YYYYYYYYYYRC,YYYYYYYYYRYG
(s,1):
WWWSSWWWWSWA,WWSSWWWWSWWA,WSSWWWWSWWWA,SSWWWWSWWWWC,SWWWWSWWWWSC,
RRRYRRRRRYYA,RRYRRRRRYYRA,RYRRRRRYYRRA,YRRRRRYYRRRC,RRRRRYYRRRYC,
WWWWSWWWWSSG,WWWSWWWWSSSA,WWSWWWWSSSWA,WSWWWWSSSWWA,SWWWWSSSWWWC,
RRRRYYRRRYYG,RRRYYRRRYYRA,RRYYRRRYYRRA,RYYRRRYYRRRA,YYRRRYYRRRRC,
WWWWSSSWWWSC
YRRRYYRRRRYC

(1,2):
WWSSSWWWSSWA,WSSSWWWSSWWC,SSSWWWSSWWSA,SSWWWSSWWSWA,SWWWSSWWSWWC,
RRYYRRRRYYYA,RYYRRRRYYYRC,YYRRRRYYYRYA,YRRRRYYYRYRA,RRRRYYYRYRRC,
WWWSSWWSWWSG,WWSSWWSWWSSA,WSSWWSWWSSWC,SSWWSWWSSWSA,SWWSWWSSWSWA,
RRRYYYRYRRYG,RRYYYRYRRYRA,RYYYRYRRYRRC,YYYRYRRYRRYA,YYRYRRYRRYRA,
WWSWWSSWSWWC
YRYRRYRRYRRC

(1,t):
WSSSWWWSSWWG,WSSSWWWSSWWC,SSSWWWSSWWSA,SSWWWSSWWSWC,SWWWSSWWSWSC,
RRYYRRRRYYYG,RYYRRRRYYYRC,YYRRRRYYYRYA,YRRRRYYYRYRC,RRRRYYYRYRYC,
WWWSSWWSWSSG,WWSSWWSWSSSA,WSSWWSWSSSSWC,SSWWSWSSSSWSA,SWWSWSSSSWSWC,
RRRYYYRYRYYG,RRYYYRYRYYRA,RYYYRYRYYRRC,YYYRYRYYRRYA,YYRYRYYRRYRC,
WWSWSSSWSWSC,
YRYRYYRRYRYC,

(2,s):

```
WWWSSWSWWSWA,WWSSWSWWSWWA,WSSWSWWSWWWA,SSWSWWSWWWWA,SWSWWSWWWWC,
YRRYRRYRRYYA,RRYRRYRRYYRA,RYRRYRRYYRRA,YRRYRRYYRRRA,RRYRRYYRRRRC,
WSWWSWWWWWSG,SWWSWWWWWSSA,WWSWWWWWSSWA,WSWWWWWSSWWA,SWWWWWSSWWWA,
RYRRYYRRRRYG,YRRYYRRRRYRA,RRYYRRRRYRRA,RYYRRRRYRRRA,YYRRRRYRRRRA,
WWWWWSSWWWWC
YRRRRYRRRRRC

(2,t):
SWWSSWSWWSWA,WWSSWSWWSWWC,WSSWSWWSWWSA,SSWSWWSWWSWC,SWSWWSWWSWSC,
YRRYRRYRRYYA,RRYRRYRRYYRC,RYRRYRRYYRYC,YRRYRRYYRYRC,RRYRRYYRYRYC,
WSWWSWWSWSSG,SWWSWWSWSSSA,WWSWWSWSSSWC,WSWWSWSSSWSA,SWWSWSSSWSWC,
RYRRYYRYRYYG,YRRYYRYRYYRA,RRYYRYRYYRRC,RYYRYRYYRRYA,YYRYRYYRRYRC,
WWSWSSSWSWSC
YRYRYYRRYRYC
```

Ponieważ równocześnie tworzone są ciągi nad alfabetem z literami W, S oraz R, Y, bardzo łatwo jest przypisać poprawną ostatnią literę każdego nowo utworzonego łącznika, według schematu:

W oraz $R = A$

S oraz $Y = C$

S oraz $R = G$

W oraz $Y = T$

Na przykład, zanim rozszerzono elementy zbioru Z tworząc łączniki, pierwsze ciągi ścieżki łukowej (s,1) to odpowiednio:
$e_1 = WWSSWWWWSWWA$ oraz $e_2 = RRYRRRRRYYRA$. Łączniki między odpowiednikiem wierzchołka s ($WWWWSSWWWWST$ / $RRRRYRRRRRYT$), z którego ta ścieżka wychodzi, a nimi, to $WWWSSWWWWSWW$ oraz $RRRYRRRRRYYR$ (stworzone zostały one poprzez skopiowanie środkowych $l-2$ liter wierzchołka s ($WWWSSWWWWS$ oraz $RRRYRRRRRY$ oraz dołączenia do nich trzeciej i drugiej litery od końca ciągów e_1 i e_2 : WW oraz YR. Teraz pozostaje jeszcze zamiana ostatnich liter w obu ciągach. Wiadomo, że są one odpowiednikiem tego samego łącznika zapisanego nad różnymi alfabetami. Z powyższej reguły widać, że kombinacja liter W oraz R daje literę A, tak więc otrzymujemy: $WWWSSWWWWSWA$ oraz $RRRYRRRRRYYA$. Aby otrzymać instancję problemu BSBH-nes należy tylko usunąć powtarzające się elementy w rozszerzonym zbiorze Z.

Dwa ciągi W/S i R/Y podane poniżej są tylko hipotetycznymi superciągami składowymi uogólnionego superciągu, ułatwiającymi zrozumienie przykładu. Stosowanie reguły

nakładania Binary powoduje redukcję liter W, S, R, Y do alfabetu {A, C, G, T} w procesie konstrukcji jednego uogólnionego superciągu. Tutaj jednak zostały one rozpisane osobno:

```
SSSSSSSSSSSG WWWWSSWWWWST WWWSSSWWWSST WSWWSSWSWWST WSWSSSWSWSST SWSWSWSWSSSG
YRYRYRYRYRYG RRRRYRRRRRYT RRRYYRRRRYYT RYRRYRRYRRYT RYRYYRRYRYYT YYYYYYYYYRYG
--------------------------------------------------------------------------------
CGCGCGCGCGCG AAAACGAAAACT AAACCGAAACCT ACAACGACAACT ACACCGACACCT CTCTCTCTCGCG
 G G G G G G  A A G A A T  A C G A C T  C A G C A T  C C G C C T  T T T T G G
```

Superciąg wypisany najniżej w przykładzie stanowi rozwiązanie dla przykładu grafu rozpatrywanego w tej sekcji. Zbudowany jest on tylko z liter na parzystych pozycjach w uogólnionym superciągu będącym rozwiązaniem dla qBSBH-ped (trzeci od góry na przykładzie). Ten z kolei uogólniony superciąg jest wynikiem nałożenia się na siebie superciągów pierwszego i drugiego w przykładzie, zgodnie z definicją nakładania dla ciągów Binary (w rzeczywistości jest tworzony z kolejnych elementów od razu, tylko na potrzeby przykładu pokazano jego oddzielne dwa superciągi Binary). Kolejne $|V|$ elementów licząc od drugiego (dla ostatniego pokazanego w przykładzie superciągu), wszystkie o długości l, stanowi odpowiedniki wierzchołków grafu, które kodują w nim ścieżkę Hamiltona. $A\ A\ G\ A\ A\ T$ stanowi odpowiednik wierzchołka s, $A\ C\ G\ A\ C\ T$ odpowiednik wierzchołka 1, itd. Cały superciąg jest odpowiednikiem ścieżki Hamiltona z przykładu.

Należy się teraz zastanowić nad warunkami poprawności całej transformacji. Podzielmy ją dla ułatwienia na trzy fazy:

1. Faza tworzenia odpowiedników wierzchołków i łuków w grafie oraz ciągów dodatkowych (początkowych i końcowych), zbudowanych nad alfabetem {A, C, G, T}.
2. Faza kodowania powyższych ciągów na alfabety {W, S, A, C, G, T} oraz {R, Y, A, C, G, T}, w celu otrzymania ciągów typu Binary.
3. Faza tworzenia łączników ciągów typu Binary, w celu otrzymania instancji problemu pozytywnego qBSBH-ned.

Rozważmy najpierw fazę pierwszą. Zostanie tutaj przytoczony wywód logiczny z artykułu [10], uzasadniający poprawność transformacji dla problemu klasycznego SBH z błędami pozytywnymi. Faza pierwsza jest bowiem tą właśnie cytowaną w rozprawie transformacją z powyższego artykułu (z drobną zmianą ostatniej litery superciągu). Elementy będące błędami pozytywnymi tworzone są w tej fazie tylko, jeśli z danego odpowiednika wierzchołka grafu G wychodzi więcej niż jedna ścieżka (łuk w grafie G). Jeśli chodzi o prawidłowe rozwiązanie to ma ono dość usystematyzowaną postać: *GGGGGG*

na początku, za nim odpowiednik wierzchołka s: $AAGAAT$, $TTTTGT$ na końcu superciągu, przed nim odpowiednik wierzchołka t: $CCGCCT$. Pomiędzy odpowiednikami s i t znajdują się odpowiedniki 6-literowe (w przykładzie) dla wierzchołków grafu w kolejności, która tworzy w tym grafie ścieżkę Hamiltona pomiędzy wierzchołkami s i t.

Załóżmy w tym momencie (w fazie I), że ścieżka w grafie istnieje między wierzchołkiem s a t. Jeden element odpowiadałby każdemu wierzchołkowi w grafie, a każde $l-1$ elementów odpowiadałoby każdemu łukowi pomiędzy wierzchołkami grafu (tzw. ścieżki łukowe). Konstrukcja elementów pozwala otrzymać ciąg $l|V|$ liter (jeżeli $l(|V|-1)+1$ ciągów jest maksymalnie na siebie nałożonych w odpowiedniej kolejności). Po dodaniu wszystkich elementów startowych i końcowych otrzymujemy ciąg $l(|V|+1)+1$ różnych elementów, długość superciągu to $l(|V|+2)$ znaków [10]. Tyle też wynosi długość superciągu pokazanego jako najniższy z czterech w przykładzie wypisanym powyżej.

Teraz załóżmy, że superciąg o długości $l(|V|+2)$ znaków istnieje i został zbudowany z $l(|V|+1)+1$ różnych elementów spektrum fazy pierwszej transformacji. Elementy muszą być na siebie maksymalnie nałożone na długości $l-1$. Jest to możliwe tylko wtedy, jeżeli między każdą parą elementów będących odpowiednikami wierzchołków istnieje $l-1$ elementów odpowiadających łukowi między takimi wierzchołkami w grafie. Gdyby próbować utworzyć superciąg tylko z odpowiedników wierzchołków i łuków, jego długość byłaby równa $|V|+(|V|+1)(l-1)$ ponieważ jest tylko $|V|$ wierzchołków. Taka rekonstrukcja miałaby mniej elementów niż jest to wymagane. Dlatego wprowadzone zostały elementy startowe i końcowe superciągu. Zmuszają one pierwszy odpowiednik wierzchołka do bycia odpowiednikiem wierzchołka startowego s, natomiast ostatni - t. Wszystkie pozostałe elementy o długości l w superciągu odpowiadają innym wierzchołkom grafu pomiędzy s a t [10]. Analizowana prawidłowa sekwencja ma budowę:

l elementów "startowych";

element będący odpowiednikiem s;

$l-1$ elementów odpowiadających łukowi wychodzącemu z s;

inne elementy reprezentujące wierzchołki i łuki w grafie;

$l-1$ elementów odpowiadających łukowi wchodzącemu do t;

element będący odpowiednikiem wierzchołka t;

l elementów "końcowych".

Uporządkowanie elementów superciągu, który można utworzyć ze spektrum tylko z fazy pierwszej całej transformacji odpowiada ścieżce Hamiltona w problemie DHPBTV.

Jak łatwo zauważyć, cała faza pierwsza jest podobna do transformacji użytej dla problemu chipu typu Gapped z błędami pozytywnymi (z wyłączeniem tamtejszych elementów nieklasycznych). Jest to prawie idealnie przytoczona transformacja z artykułu [10], ponieważ stanowi ona punkt wyjścia dla dalszych faz.

Celem faz kolejnych: II i III jest otrzymanie instancji dla problemu Binary.

Superciąg zbudowany z elementów takiej w pełni już przetransformowanej instancji problemu qBSBH-ped istnieje wtedy i tylko wtedy, gdy istnieje ścieżka Hamiltona w grafie z instancji problemu DHPBTV. Z superciągu takiego będzie można otrzymać superciąg opisany przed chwilą tylko dla fazy I, a co za tym idzie - będzie on rozwiązaniem dla problemu ścieżki Hamiltona. Należy uzasadnić, dlaczego II i III faza transformacji nie zmieniają warunków poprawności.

Jeżeli chodzi o fazę II sytuacja jest dość jasna. Kodowanie binarne tworzy dwa zestawy ciągów znaków dla zbiorów S_{B1} i S_{B2}, o długościach elementów dwukrotnie większych niż elementy z fazy I. Są to jednak wciąż unikalne odpowiedniki elementów fazy I, powtarzają się wtedy i tylko wtedy, jeśli powtarzają się elementy w tejże pierwszej fazie. Aby mogły zbudować superciąg będący odpowiednikiem tego przed chwilą opisanego, należałoby wymusić na algorytmie nakładanie tylko i wyłącznie na $l_2 - 2$ znakach, gdzie $l_2 = 2l$ to długość elementów po przekodowaniu w fazie II. Poszukiwany superciąg musiałby mieć długość $l_2(|V| + 2)$ znaków (lub $2l(|V| + 2)$), składałby się z $l(|V| + 1) + 1$ elementów zbioru S_{B1} i tylu samo elementów S_{B2} (zbiory te tworzyłyby wtedy pewne spektrum po usunięciu powtórzeń ze zbioru Z). Nakładanie elementów odbywałoby się zgodnie z regułą nakładania dla ciągów typu Binary.

Po fazie II wciąż nie dysponujemy jednak spektrum dla błędów pozytywnych. Brak wspomnianych w transformacji "łączników" powoduje, że w tym momencie dysponujemy pewnym spektrum przejściowym, z obydwoma rodzajami błędów. Dlatego w fazie III transformacji tworzone są łączniki, których obecność eliminuje całkowicie błędy negatywne, pozwalając otrzymać pełną instancję z błędami pozytywnymi. Aby zakończyć wywód o poprawności transformacji należy określić dwie zasadnicze rzeczy:

1) liczbę elementów potrzebną do rekonstrukcji uogólnionego superciągu z obu zbiorów Binary;

2) warunek, w myśl którego tworzone łączniki nie będą się powtarzać w rozwiązaniu, tj. uogólnionym superciągu.

Określenie liczby elementów pełnego spektrum (z łącznikami, po usunięciu powtórzeń ze zbioru Z) jest sprawą prostą. Należy zrekonstruować ciąg o długości $n = l_2(|V| + 2)$ znaków, używając elementów o długości l_2 z maksymalnym nałożeniem. Z rozwinięcia wzoru $n - l_2 + 1$ otrzymujemy $l_2(|V| + 1) + 1$ elementów dla zbioru S_{B1} i tyle samo dla S_{B2}.

Bardziej skomplikowane jest wykazanie, że łączniki nie powtarzają się w ramach swojego podzbioru, oraz żaden z nich nie powtarza się z elementami budującymi superciąg-rozwiązanie otrzymanymi zaraz po II fazie transformacji. Rozważmy poniższy superciąg składowy:

```
SSSSSSSSSSSS WWWWSSWWWWSW WWWSSSWWWSSW WSWWSSWSWWSW WSWSSSWSWSSW SWSWSWSWSSSS
------------------------------------------------------------------------------
 G G G G G G  A A G A A T  A C G A C T  C A G C A T  C C G C C T  T T T T G G
                G     T      G     T      G     T      G     T (motyw)
```

Jak napisano w opisie II fazy transformacji, litery z {A, C, G, T} kodowane są według zasad:

litera A zmieniana jest na WW i RR

litera C zmieniana jest na WS i RY

litera G zmieniana jest na SS i YR

litera T zmieniana jest na SW i YY

Jak widać odpowiednik wierzchołka s czyli $A\ A\ G\ A\ A\ T$ zakodowany został jako $WWWWSSWWWWST$. O ile na alfabecie binarnym trudno dostrzec jakieś specyficzne wzorce, o wiele łatwiej jest je zauważyć na słowach alfabetu {A, C, G, T}. Poza pierwszym i ostatnim elementem, które należą do specyficznego podzbioru elementów startowych i końcowych ($GGGGGG$ i $TTTTGG$), litery G i T nigdy nie występują obok siebie. Dodatkowo są one zawsze oddalone od siebie o stałą długość (dwóch liter w powyższym przykładzie). Warunek ten spełniają dowolne przesunięcia "okna" o rozmiarze 6 celem analizy wszystkich ciągów 6-literowych zaczynając od odpowiednika wierzchołka s, kończąc na odpowiedniku t. Na przykład element $A\ A\ T\ A\ C\ G$ powstały z trzech ostatnich liter odpowiednika wierzchołka s i trzech pierwszych liter odpowiednika wierzchołka 1 (odpowiednio: $A\ A\ G\ A\ A\ T$ oraz $A\ C\ G\ A\ C\ T$) posiada tylko jedną literę T i tylko jedną G, które są oddalone od siebie o 2 znaki. Dla dowolnego przesunięcia (poza elementem startowym i końcowym $GGGGGG$ i $TTTTGG$) reguła ta jest spełniona.

Rozpatrzmy teraz jaki ciąg powstanie z "łączników" otrzymanych w III fazie transformacji. W tym celu usuniemy pierwszą literę S superciągu powyższego i jego ostatnią literę S. Otrzymamy superciąg krótszy o 2 znaki, który dekodujemy na alfabet 4-literowy:

```
SSSSSSSSSSSW WWWSSWWWWSWW WWSSSWWWSSWW SWWSSWSWWSWW SWSSSWSWSSWS WSWSWSWSSS
------------------------------------------------------------------------------
 G G G G G T  A C T A C A  A G T A G A  T C T T C A  T G T T G C  C C C C C G
                T     A      T     A      T     A      T     C (motyw)
```

Jak widać zmienił się motyw elementów: zamiast kombinacji G-T, mamy teraz kombinację T-A. Jest ona stała, tj. dla dowolnego grafu, dowolnej ścieżki Hamiltona zawsze powstanie, mając stałą długość rozdzielenia liter motywu - przykładzie rozpatrywanym są to 2 znaki. Dzieje się tak, ponieważ za jej tworzenie odpowiada kombinacja dwóch liter W/S alfabetu binarnego kodującego litery motywu G i T z poprzedniego superciągu (częściowo, ponieważ mówimy o superciągu przesuniętym o 1 znak). Uniemożliwia ona powtarzalność elementów "łączników" ze sobą i z innymi elementami superciągu-rozwiązania. Dowolny element 6-literowy tego superciągu łączników nie będzie w stanie utworzyć motywu G-T występującego tylko raz, ponieważ zawsze po stałej liczbie znaków po T występować będzie litera A. Nawet jeśli motyw G-T byłby w stanie powstać z innych 2-literowych kodów znaków, oznaczać to będzie min. 2 litery T w 6-znakowym słowie. A takie słowo nie może przecież wystąpić w superciągu opisanym przy I fazie transformacji pomiędzy odpowiednikami wierzchołków s i t. Stąd widać, że utworzenie łączników dla alfabetu W/S zachowuje unikalność elementów przy założeniu ich obecności w prawidłowym superciągu-rozwiązaniu. Dla alfabetu R/Y sytuacja wygląda następująco:

```
RYRYRYRYRYRR RRRYRRRRRYYR RRYYRRRRYYYR YRRYRRYRRYYR YRYYRRYRYYYY YYYYYYYYRY
------------------------------------------------------------------------------
 C C C C C A  A C A A C G  A T A A T G  G C A G C G  G T A G T T  T T T T C
                A     G      A     G      A     G      A     T (motyw)
```

Tutaj motyw elementów odpowiedników ścieżki od s do t to A-G. Podobnie jak poprzednio, jego obecność uniemożliwia utworzenie elementów na alfabecie R/Y które miałyby motyw oryginalny G-T. Nawet jeśli pojawi się on w 6-literowym słowie, będzie ono wtedy zawierało minimum 2 litery G, czyli będzie to słowo niemożliwe do uzyskania w oryginalnym superciągu (z I fazy transformacji) pomiędzy odpowiednikami wierzchołków s i t. To wykazanie unikalności "łączników" (po zakodowaniu ich oczywiście na odpowiednie alfabety binarne) kończy ostatnią część poprawności transformacji. Łączniki nie powtarzają się też przy tworzeniu ich dla elementów startowych i końcowych (przed s i po t) - te elementy są tworzone w specyficzny sposób i są całkowicie niezależne od struktury ścieżki Hamiltona w dowolnym grafie. Z powyższych dwóch przykładowych obrazków widać, że łączniki zachodzące na te elementy tworzą 6-literowe słowa które również nie występują w superciągu fazy I.

Podsumowując można powiedzieć, że:

1. Pierwsza faza transformacji spełnia wszystkie wymogi poprawności opisane w artykule [10] (w rzeczywistości jest to bowiem użyta tam właśnie transformacja dla problemu nazwanego $qSBH - epd$).

2. Faza druga przekodowuje ciągi znaków z fazy I nad alfabetami problemu Binary, zachowując unikalność elementów.

3. Faza trzecia eliminuje błędy negatywne, umożliwiając powstanie spektrum problemu pozytywnego quasi-BSBH. Nie powoduje ona powstania powtarzających się elementów wchodzących w skład rozwiązania, tj. uogólnionego superciągu o wyznaczonej długości.

4. Uogólniony superciąg zbudowany z precyzyjnie określonej liczby elementów zbiorów S_{B1} i S_{B2} umożliwia odczytanie sekwencji znaków, która koduje odpowiedniki ścieżki Hamiltona na alfabecie 4-literowym wtedy i tylko wtedy, jeśli taka ścieżka istnieje w grafie istancji problemu DHPBTV od którego transformację rozpoczęto.

Prowadzi to do udowodnienia silnej NP-zupełności problemu qBSBH-ped.

Twierdzenie 7.4.2. Problem BSBH-pes (w wersji przeszukiwania) jest silnie NP-trudny.

Dowód
Dokładnie tak samo jak w dwóch poprzednich rozdziałach, uzasadnienie powyższego twierdzenia wynika z faktu, że gdyby dla problemu przeszukiwania BSBH-pes istniał algorytm wielomianowy, to można by go wykorzystać do rozwiązania w wielomianowym czasie problemu quasiBSBH-ped. Byłoby tak dlatego, że zbiór wszystkich instancji problemu przeszukiwania stanowi zbiór wszystkich instancji z odpowiedzią TAK problemu qBSBH-ped. Z tego powodu można by użyć tego algorytmu dla każdej instancji problemu quasi. Jeżeli w wielomianowym czasie algorytm nie znajdzie rozwiązania, odpowiedź dla qBSBH-ped brzmiałaby NIE, w przeciwnym wypadku byłaby to odpowiedź TAK (jeśli algorytm znajdzie rozwiązanie w wielomianowym czasie). W świetle teorii złożoności oraz założenia, że $P \neq NP$ nie jest to możliwe. □

7.5 Algorytm dla chipu typu Binary

W tej części niniejszego rozdziału zostanie opisany zaproponowany algorytm dla chipu typu Binary. W pierwszej kolejności zostaną omówione cele jakie przyświecały konstrukcji algorytmu, a także dane wejściowe. Opis ten uwzględnia także proces tworzenia grafów z elementów spektrum. Opis algorytmu wraz z przykładami oraz pseudokodem ilustrującym poszczególne jego procedury znajdzie się w kolejnym podrozdziale. W celu łatwiejszego przedstawienia zaproponowanego algorytmu zbiory S_{B1} i S_{B2} (notacja używana w sekcji dotyczącej złożoności obliczeniowej) będą zastępowane odpowiednio oznaczeniami S_{WS} oraz S_{RY}. Jak łatwo się domyślić, oznaczają one zbiory spektrum z elementami zbudowanymi odpowiednio nad alfabetem {W, S, A, C, G, T} oraz alfabetem {R, Y, A, C, G, T}.

7.5.1 Dane wejściowe

Celem algorytmu jest rekonstrukcja sekwencji DNA o określonej długości n ze spektrum DNA otrzymanego przy użyciu nieklasycznego chipu typu Binary. Opracowany algorytm radzi sobie z występowaniem wszystkich rodzajów błędów hybrydyzacji, zarówno pozytywnych jak i negatywnych. W przypadku tych drugich, mogą wynikać one zarówno z błędów procesu odczytu sond chipu, jak i z powtórzeń podłańcuchów w sekwencji DNA. Dane wejściowe algorytmu są następujące:

1) spektrum DNA z eksperymentu hybrydyzacyjnego, zawierające elementy pochodzące z obu połówek chipu typu Binary: S_{WS} oraz S_{RY};

2) długość n badanego fragmentu DNA;

3) parametr k, określający długość oligonukleotydów chipu;

4) informacja o sondzie w każdej z dwóch części chipu, która hybrydyzowała z początkowym fragmentem DNA (o długości $k+1$ nukleotydów).

Parametry n oraz k pozwalają obliczyć wiele wartości liczbowych niezbędnych dla dalszej pracy algorytmu. Parametr k określa długość oligonukleotydów w sondach, zgodnie ze wzorem (7.1). Mając daną długość DNA n, możliwe jest określenie liczby elementów obu zbiorów tworzących spektrum idealne (wzór (7.3)). Gdyby możliwe było wykluczenie któregoś z typów błędów hybrydyzacji, możliwe stałoby się dokładne określenie liczby błędów hybrydyzacji w danym spektrum. W innym wypadku algorytm określa tę liczbę w sposób przybliżony, tak jak było to realizowane dla algorytmów z poprzednich rozdziałów.

Spektrum DNA jest podzielone na dwa zbiory: zbiór S_{WS} oraz zbiór S_{RY}, zawierające elementy zbudowane nad 6-literowymi alfabetami, odpowiednio {W, S, A, C, G, T} oraz {R, Y, A, C, G, T}, które posłużą do budowy dwóch grafów skierowanych. W dalszej części rozdziału określenia "ciąg" i "wierzchołek" (a także "następnik") używane będą zamiennie, ponieważ każdy wierzchołek grafu jest opisany przez pewien ustalony ciąg znaków, tj. ciąg typu Binary. Zbiory spektrum służą do budowy dwóch różnych grafów, zgodnie z zasadami danymi poniżej:

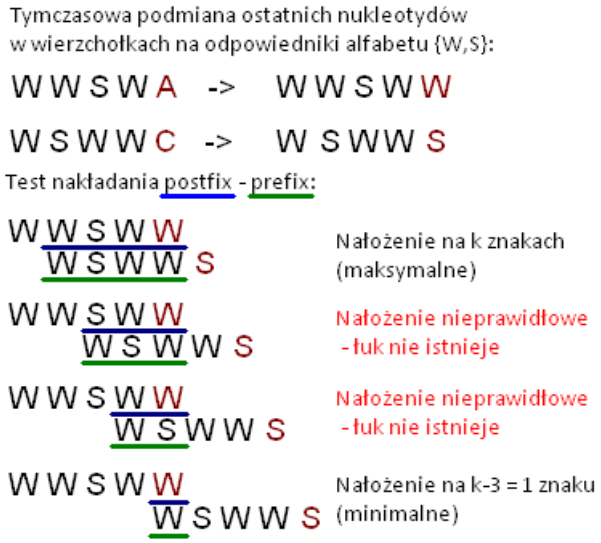
1) wszystkie elementy zbiorów S_{WS} oraz S_{RY} są traktowane jako wierzchołki grafów.

2) łuki w każdym grafie tworzone są na bazie nakładania na siebie postfixu i prefixu dwóch wierzchołków na długości od 1 do k znaków.

Postfixem o długości l_{post} ciągu e nazywamy podciąg jego ostatnich l_{post} liter, przy czym długość maksymalna postfixu jest równa $l_{post}(max) = k$ ostatnich liter, a minimalna długość to 1.

Prefixem o długości l_{pref} ciągu e nazywamy podciąg jego początkowych l_{pref} liter, przy czym długość maksymalna prefixu jest równa $l_{pref}(max) = k$ początkowych liter, a minimalna długość to 1.

Dwa ciągi e_1 i e_2 nakładają się na siebie na długości r, jeżeli postfix ciągu e_1 o długości r jest identyczny z prefixem ciągu e_2 o tej samej długości (zgodnie z regułami nakładanie się ciągów typu Binary)). Podane tutaj definicje ilustruje przykład z rysunku 7.2:



RYSUNEK 7.2: Przykład podmiany ostatniego elementu oraz wszystkie istniejące prawidłowe i nieprawidłowe nałożenia wierzchołków

W części dotyczącej złożoności obliczeniowej problemów dla chipu typu Binary, określono już regułę nakładania się ciągów typu Binary. Reguła ta została w przykładzie zastąpiona tymczasową podmianą ostatniego znaku opisującego sondę. Na rysunku 7.2 przedstawiono przykład dla grafu budowanego z elementów S_{WS}, analogicznie przebiega procedura dla grafu z elementów zbioru S_{RY}. Jak wiadomo z opisu chipu, na końcu każdego elementu spektrum znajduje się litera z alfabetu {A, C, G, T}. Na rysunku wierzchołki ze zbioru S_{WS} zostają czasowo podmienione na swoje odpowiedniki, z alfabetu binarnego {W, S}. Dzięki temu o wiele prościej porównać ze sobą prefix i postfix dwóch wierzchołków w procesie wyszukiwania łuków między nimi.

Podmiana ostatniego elementu jest tymczasowa i dotyczy tylko procedury budowania łuków w grafie. Algorytm szukający ścieżki z wierzchołków grafu ma pełne informacje o danym wierzchołku, tj. także o tym, jaki naturalny nukleotyd występuje na końcu łańcucha opisującego dany wierzchołek. Jest to niezbędne do "synchronizacji" obu grafów

(zbudowanych z elementów dwóch zbiorów spektrum BSBH) w procesie przeszukiwania. Idea tego mechanizmu przypomina łączenie w poprzednim rozdziale, dla chipu Alternating, ścieżek nukleotydów nieparzystych i parzystych przez elementy drugiego zbioru spektrum ASBH. Łuki tworzone są tylko w wypadku prawidłowych nałożeń, na przykładzie z rysunku 7.2 są to: nałożenie maksymalne ($l_{overlap} = k = 4$, waga łuku $w(k) = 1$), oraz minimalne ($l_{overlap} = k - 3 = 1$ dla $k = 4$, waga łuku $w(k) = 4$).

Procedura tworzenia drugiego grafu jest identyczna, ponieważ jedyną zmianą jest użycie w 6-literowym alfabecie znaków R i Y zamiast W i S.

7.5.2 Zasada działania algorytmu

Opracowany algorytm jest algorytmem dokładnym, starającym się przeszukać całą przestrzeń rozwiązań. Z uwagi jednak na aspekty praktyczne, jak przede wszystkich czas obliczeń, działanie algorytmu sprowadza się do przeszukania tylko jej części. Algorytm, poza najważniejszymi danymi wejściowymi z eksperymentu hybrydyzacyjnego, posiada pewne regulowane wartości parametrów swojej pracy. Możliwe jest takie ich ustawienie, aby algorytm przeszukiwał całą dostępną przestrzeń rozwiązań. Z uwagi na jej rozmiar w przypadku większości praktycznie występujących spektr dla problemu BSBH, algorytm ogranicza jednak przeszukiwanie tylko dla pewnego dostępnego czasu obliczeń, liczby wewnętrznych iteracji czy znalezionych rozwiązań niejednoznacznych w wyznaczonym czasie. Parametry te zostaną opisane dalej w niniejszym podrozdziale, w części dotyczącej warunków zatrzymania przeszukiwania.

Podobnie jak w poprzednim rozdziale (tj. w przypadku algorytmu chipu typu Alternating), tutaj również działanie algorytmu polega na budowaniu ścieżek w dwóch grafach. Jednakże w przeciwieństwie do podejścia stosowanego dla poprzedniego chipu, jeden krok elementarny głównej pętli powiększa ścieżki poprzez dodanie nowego wierzchołka *równocześnie* w obu grafach. Jest to podyktowane potrzebą wzajemnej weryfikacji wierzchołków względem siebie, tj. zgodności ostatniego nukleotydu. Na potrzeby dalszych rozważań przyjmowane jest, że ścieżka na grafie o wierzchołkach zbudowanych nad alfabetem {W, S, A, C, G, T} będzie oznaczana przez P_{WS}, druga ścieżka natomiast przez P_{RY}.

Pseudokod głównej pętli algorytmu przedstawia się następująco 6:

Główna pętla algorytmu pozostaje niezmienna niezależnie od rodzaju użytego chipu. Procedury przygotowania grafów są różne w zależności od użytego chipu, a wewnętrzne mechanizmy kontrolujące zachowanie całego algorytmu również mają różne wartości, w zależności od zastosowanego podejścia (Gapped, Alternating, Binary). Pętla główna zostanie tutaj przypomniana, natomiast dalej, szczegółowo opisane zostaną procedury przez nią wywoływane, specyficzne dla omawianego algorytmu i problemu przez niego

Algorithm 6 Pseudokod głównej pętli algorytmu dla chipu typu Binary.

```
1: while (Time < Limit AND s_space ≠ empty AND Stop = FALSE) do
2:     Candidates ← addOutgoingVerticesToList(Current_Vertex)
3:     Success_Flag ← addNewVertexToSolution(Candidates);
4:     if (Success_Flag = FALSE) then
5:         if (Path_Length = MaxValue) then
6:             Ok = testSolution(Solution);
7:             if (Ok = TRUE) then
8:                 addToSolutionList(Solution)
9:                 reverseSteps(Solution)
10:            else
11:                reverseSteps(Solution)
12:            end if
13:        else
14:            reverseSteps(Solution)
15:        end if
16:    end if
17: end while
```

rozwiązywanego.

W linii 1 sprawdzane są trzy główne warunki zatrzymania pracy algorytmu. Są to kolejno: limit czasu pracy, wyczerpanie się przestrzeni rozwiązań oraz ogólna flaga stopu, ustawiana w zależności od stanów końcowych poszczególnych dalszych procedur algorytmu. Warunki zatrzymania przeszukiwania grafów są następujące:

1) sprawdzono całą przestrzeń rozwiązań;

2) koniec przestrzeni rozwiązań;

3) osiągnięto limit rozwiązań dodanych do listy (limit rozwiązań niejednoznacznych);

4) osiągnięto limit rekonstrukcji par ścieżek o zadanej długości, które następnie zostały odrzucone przez procedurę weryfikacji rozwiązań.

Celem każdego kroku pętli jest dodanie kolejnej pary następników, po jednym dla każdego grafu. Wybór par kandydatów na następników odbywa się w linii 2, są one umieszczane na liście *Candidates*. W linii 3 algorytm stara się dodać pierwszą wolną (z listy) parę wierzchołków do ścieżek w grafach. Jeśli się to nie uda, sprawdzane są powody. Jeżeli powstały już ścieżki o wielkości umożliwiającej rekonstrukcję DNA o długości n, jest ono odtwarzane ze złożenia obu ścieżek, sposobem który opisano na początku niniejszego rozdziału. Rozwiązanie jest też weryfikowane pod względem liczby

wierzchołków użytych z obu części spektrum z uwzględnieniem wpływu błędów hybrydyzacji. Zaakceptowane rozwiązania są dodawane do listy rozwiązań. Linie 9, 11 oraz 14 wywołują procedurę powrotu do wierzchołków poprzednich, celem konstrukcji innych ścieżek. Następuje to odpowiednio, w przypadku dodania rozwiązania do listy (linia 9), odrzucenia danej pary ścieżek jako rozwiązania dopuszczalnego (linia 11), a także w przypadku, w którym algorytm musi się wycofać, ponieważ z danej pary wierzchołków nie ma możliwości przejścia do akceptowalnej pary następników (linia 14).

Teraz omówiona zostanie procedura z linii 2 głównej pętli algorytmu, odpowiedzialna za tworzenie listy par potencjalnych kandydatów na następniki w obu grafach. Szczególna uwaga będzie poświęcona zawartemu w niej mechanizmowi wzajemnej weryfikacji wierzchołków obu grafów. Psedokod tej części algorytmu jest następujący:

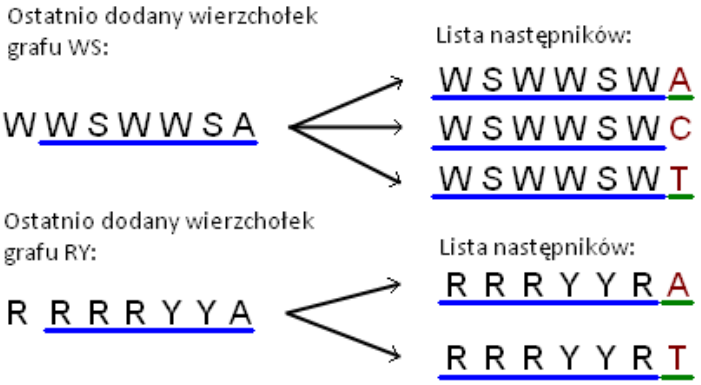
Algorithm 7 Pseudokod poszukiwania par następników dla ścieżek w grafach WS/RY.

```
1: for (all_overlap_values_starting_from_maximum) do
2:     for (VertexWS ← OverlapSet) do
3:         for (VertexRY ← OverlapSet) do
4:             if (sameLastNucleotide(VertexWS, VertexRY) = TRUE) then
5:                 Candidates ← addPair(VertexWS, VertexRY);
6:             end if
7:         end for
8:     end for
9: end for
```

Na początku algorytmu 7 znajdują się trzy pętle. Pierwsza z nich ustala aktualnie testowany rozmiar nakładania, zaczynając od maksymalnego, czyli na k znakach (linia 1). W linii 2 procedura wybiera pierwszy następnik dla ostatnio dodanego wierzchołka ścieżki P_{WS}, który nakłada się na nim na dokładnie tylu znakach, ile ustalono w danym kroku pierwszej pętli. W linii 3 wybierany jest pierwszy następnik o ustalonej w 1 linii długości nałożenia na ostatni wierzchołek ścieżki P_{RY}. W linii 4 para następników jest sprawdzana. Testowana jest identyczność ostatnich nukleotydów będących literami z alfabetu {A, C, G, T} w obu wybranych następnikach. Jeżeli są one takie same, para następników dodawana jest do listy (linia 5). Następnie rozpatrywana jest para składającą się z kolejnego następnika ścieżki P_{RY} (o ile istnieje) oraz (nadal) z pierwszego następnika ścieżki P_{WS}, ustalonego w linii 2. Procedura jest kontynuowana dla wszystkich kombinacji par wierzchołków (tj. następników) o ustalonym maksymalnym nałożeniu, które następnie jest w linii 1 zastępowane wartością nałożenia o 1 mniejszą podczas kolejnego kroku tej pętli. Następnie wybierane i sprawdzane są wszystkie kolejne pary istniejących następników dla danego, nowego nałożenia. W procedurze tworzenia listy par następników zawarta jest więc jedyna dostępna w tym chipie forma weryfikacji wierzchołków.

Zapewnia ona, że w obu grafach znaleziona ścieżka odpowiadać będzie identycznemu fragmentowi DNA.

Przykład dla wierzchołków o maksymalnym nałożeniu w obu grafach pokazuje rysunek 7.3.



RYSUNEK 7.3: Przykład tworzenia par potencjalnych następników wraz z weryfikacją

Na przykładzie z rysunku 7.3, z ostatniego wierzchołka ścieżki grafu WS prowadzą łuki do trzech możliwych następników, natomiast tylko do dwóch w ścieżce dla grafu RY. Tylko dwie pary następników mogą być brane pod uwagę w takim wypadku, ponieważ następnik $WSWWSWC$ nie posiada odpowiednika na liście następników w RY, który miałby ten sam ostatni nukleotyd. Podobnie tworzy się pary następników z mniejszym nałożeniem.

W pętli głównej po utworzeniu listy par kandydatów, wywoływana jest procedura dodawania nowych wierzchołków do ścieżek z listy par następników. Warunki dodania par wierzchołków są podobne jak w poprzednich algorytmach. Dwa najważniejsze to sprawdzenie, czy rozszerzone ścieżki nie przekroczą maksymalnej dopuszczalnej długości oraz czy w przypadku nałożenia mniejszego niż maksymalne, powiększony o odpowiednią wartość licznik takich nałożeń wciąż znajduje się poniżej limitu ustalonego dla poszukiwanego rozwiązania.

Rozmiar przestrzeni rozwiązań jest zależny od długości DNA (przekładającej się na liczbę wierzchołków grafu) oraz liczby połączeń (łuków) między wierzchołkami. Przy maksymalnym nałożeniu, każdy wierzchołek może mieć do czterech następników. Nałożenia mniejsze powodują podwyższanie tej wartości. W algorytmie zawarty jest więc limit tego, ile razy w rekonstrukcji ścieżek można użyć nałożeń mniejszych niż maksymalne. Jest on całkowicie zależny od liczby błędów negatywnych w spektrum. W tym miejscu pojawia się jednak problem jego obliczenia. Wpływ różnych rodzajów błędów na określanie tej wartości, a co za tym idzie na działanie całego algorytmu podsumowują następujące wymienione przypadki:

1. Brak błędów hybrydyzacji w spektrum. Możliwa jest precyzyjna weryfikacja następników, tj. wiadomo, że wszystkie elementy z S_{WS} są obecne oraz mają odpowiadające im pary w części S_{RY} spektrum. Limit wierzchołków nienakładających się maksymalnie jest równy zero. Jest to niestety dość nierealistyczny przypadek. Nawet po wyeliminowaniu jakichkolwiek nieprecyzyjności procesu hybrydyzacji i detekcji sond, pozostaje problem błędów negatywnych wynikających z powtórzeń. Są one zależne tylko od budowy badanego łańcucha DNA. Wymagane byłoby tutaj bezbłędne określenie liczby powtórzeń każdego elementu spektrum w sekwencji, czego obecnie nie daje się precyzyjnie wykonać.

2. W spektrum występują tylko błędy negatywne wynikające z powtórzeń. W przeciwieństwie do poprzednich algorytmów, ten rodzaj błędów ma już pewien wpływ na proces rekonstrukcji oraz doboru par następników. Każdy taki błąd powoduje, że w częściach spektrum znajduje się mniej elementów. Z tego powodu należy używać nałożeń mniejszych niż maksymalne. Dochodzi tutaj jednak pewna dodatkowa cecha charakterystyczna zaproponowanego algorytmu dla chipu Binary. Każdy mianowicie błąd negatywny (dowolnego rodzaju) w jednej części spektrum, pociąga za sobą niemożność użycia odpowiadającego mu elementu z drugiej części spektrum. Tylko para następników może być brana pod uwagę, tak więc w przeciwieństwie do podejścia opisanego dla dwóch poprzednich chipów, gdzie błędy negatywne tego rodzaju miały wpływ głównie na mechanizm weryfikacji, tutaj istotna jest ich suma z obu częściach spektrum przy ustalaniu liczby elementów, które musi zawierać rozwiązanie dopuszczalne. Oczywiście zdarza się, że ze spektrum odpadną oba elementy tej samej pary, lecz taka sytuacja jest na tyle rzadka, że nie jest brana pod uwagę w próbach ustalenia liczby błędów. W tym przypadku za rozwiązanie dopuszczalne uważa się to, w którym ze zbioru S_{WS} użyto przynajmniej liczby elementów równej liczności tego zbioru pomniejszonej o liczbę błędów negatywnych z drugiego zbioru (S_{RY}) oraz odwrotnie: minimalna liczba elementów którą należy użyć ze zbioru S_{RY}, to jego liczebność pomniejszona o liczbę błędów negatywnych w S_{WS}. Obliczenie liczby błędów negatywnych w tym i następnym przypadku nie nastręcza jednak większych problemów. Dla każdego zbioru jest ona różnicą między licznością zbioru danego w spektrum a licznością zbioru spektrum idealnego.

3. Błędy negatywne ogólnego rodzaju, wynikające z powtórzeń oraz nieprecyzyjności w procesie hybrydyzacji. Przypadek ten w przeciwieństwie do dwóch poprzednich podejść (chipy Gapped i Alternating) jest identyczny z poprzednim opisanym tutaj dla chipu Binary. Jeżeli przez liczbę błędów negatywnych obu części spektrum oznaczymy różnicę pomiędzy liczbą ich elementów a liczbą elementów w spektrum idealnym:

$$err_{WS} = |S^{is}_{WS}| - |S_{WS}| \text{ oraz } err_{RY} = |S^{is}_{RY}| - |S_{RY}| \tag{7.10}$$

gdzie $|S^{is}_{WS}|$ i $|S^{is}_{RY}|$ to liczba elementów obu zbiorów spektrum idealnego, wtedy minimalna liczba elementów w obu zbiorach spektrum danego, którą należy wykorzystać, jest równa:

$$min_{WS} = |S_{WS}| - err_{RY} \text{ oraz } min_{RY} = |S_{RY}| - err_{WS} \tag{7.11}$$

Podsumowując przypadki 2. i 3. można powiedzieć, że w przeciwieństwie do pozostałych chipów, są one tak samo trudne. Geneza błędu negatywnego nie ma znaczenia dla zaproponowanego algorytmu chipu Binary.

4. Błędy negatywne oraz pozytywne w spektrum. Jest to najtrudniejszy przypadek, jednak najbardziej realistyczny. Nie jest możliwe określenie liczby błędów negatywnych, przez co nie można precyzyjnie określić minimalnej liczby elementów, które budują rozwiązania dopuszczalne z obu części spektrum. Liczba ta jest ustalana w przybliżeniu, zarówno aby określić limit wierzchołków z nałożeniem mniejszym niż maksymalne w rozwiązaniu oraz, aby jak poprzednio określić ile elementów z każdego zbioru spektrum musi zostać (minimalnie) wykorzystana. Przy założeniu np. 10% błędów w całym spektrum (te wartości będą zmieniane i testowane dalej w pracy w ramach eksperymentów obliczeniowych) przyjmuje się, że liczba równa 10% wielkości spektrum idealnego to limit wykorzystania wierzchołków z nałożeniem poniżej maksymalnego.

Podsumowując tematykę błędów hybrydyzacji, określenie liczby błędów negatywnych występujących w spektrum ma zasadnicze znaczenie dla efektywności przeszukiwania przestrzeni rozwiązań. Jeżeli nałożenie nowego następnika z ostatnim wierzchołkiem jest mniejsze niż maksymalne oznacza to, że pomaga on w "naprawie" efektu wystąpienia błędu negatywnego, który uniemożliwił nałożenie na równo $max_{overlap} = k$ znakach w tym miejscu. Branie pod uwagę wierzchołków o nałożeniu mniejszym niż $max_{overlap}$ znacznie powiększa przestrzeń rozwiązań omawianego problemu. Gdy możliwe jest precyzyjne określenie liczby błędów negatywnych (tj. w przypadku, gdy nie ma błędów pozytywnych), algorytm wiedząc, że "naprawił" już odpowiednią liczbę błędów, przestaje brać pod uwagę wierzchołki z nałożeniem mniejszym niż $max_{overlap}$ znaków. Niestety, najbardziej realistyczny przypadek pracy to ten, w którym występują wszystkie rodzaje błędów. Jak już wyjaśniono, w tym przypadku niemożliwe jest precyzyjne określenie liczby błędów negatywnych. Algorytm dla zachowania efektywności przyjmuje przybliżoną wartość, np. zakładając istnienie maksymalnie 10% błędów negatywnych w obu

zbiorach spektrum. Oznacza to między innymi akceptację rozwiązań, które mając ustalony rozmiar poszukiwanego DNA wykorzystały przynajmniej 80% liczby elementów spektrum idealnego. Porównanie skuteczności między takim podejściem z wartościami limitu równymi także 15% i 20%, a podejściem ze znajomością precyzyjnej liczby błędów negatywnych zostanie przedstawione dalej, w rozdziale z wynikami testów dla algorytmu.

Wracając do pseudokodu głównej pętli algorytmu, należy wciąż opisać dwie jego ważne procedury: powrót do poprzednich wierzchołków w celu zbadania innych rozgałęzień w grafach oraz dodanie rekonstrukcji DNA do listy rozwiązań. To ostatnie wiąże się z jej odtworzeniem z dwóch ścieżek P_{WS} oraz P_{RY}.

Realizacja powrotu do poprzednich wierzchołków jest taka sama jak w poprzednich przedstawionych algorytmach z uwzględnieniem oczywiście faktu jednoczesnej operacji na wierzchołkach dwóch grafów. Procedura ta jest wywoływana w liniach 9, 11 i 14 pseudokodu głównej pętli algorytmu, w przypadkach gdy:

1) nowa rekonstrukcja została utworzona ze ścieżek o odpowiedniej długości;

2) nie powiodła się próba dodania nowego wierzchołka z listy kandydatów na następników.

W takich wypadkach algorytm wraca do poprzednich wierzchołków, co jest realizowane w ten sam sposób jak w poprzednich podejściach. Jedyna różnica dotyczy tego, że w przypadku omawianego tutaj algorytmu dla chipu typu Binary wycofywana jest aktualna *para* wierzchołków w obu grafach. Pozostałe polecenia dotyczące resetowania ustawień przeszukiwania czy skracana ścieżek są identyczne jak w poprzednio opisanych algorytmach.

Przed dodaniem znalezionej sekwencji do zbioru rozwiązań musi ona zostać odtworzona w postaci ciągu nad alfabetem 4-literowym {A, C, G, T} oraz zweryfikowana. Jak zostało już napisane, złożenia liter z {W, S} oraz z {R, Y} jednoznacznie identyfikują konkretny nukleotyd ze zbioru {A, C, G, T}. Ponieważ oba ciągi znaków (odpowiedniki ścieżek) są jednakowej długości, w liniowym czasie po ich złożeniu algorytm odtwarza sekwencję DNA.

Przed dodaniem nowego rozwiązania do listy rozwiązań musi być ono zweryfikowane. Proces ten polega na testowaniu występowania elementów spektrum w zrekonstruowanej sekwencji, zgodnie z zasadami reguły nakładania dla ciągów typu Binary. Procedura ta jest zależna od przypadku występowania lub nie dwóch typów błędów hybrydyzacji. W przypadku, gdy błędy negatywne i pozytywne nie występują razem, można precyzyjnie określić liczbę elementów spektrum DNA, które muszą zostać wykorzystane. Niestety realistyczny przypadek to ten, w którym występują wszystkie rodzaje błędów hybrydyzacji

w spektrum danym na wejściu. Algorytm posługuje się wtedy przybliżoną minimalną liczbą elementów, akceptując rozwiązania, które w procesie tworzenia ścieżek zużyły liczbę elementów spektrum zależną od przyjętego limitu.

Rozdział 8

Wyniki eksperymentów obliczeniowych

8.1 Wstęp

W rozdziale tym zostaną przedstawione wyniki eksperymentów obliczeniowych, przeprowadzonych dla algorytmów zaproponowanych dla trzech opisanych w pracy chipów nieklasycznych. Dla porównania przedstawione będą także wyniki algorytmu, który symuluje ten sam schemat postępowania co trzy nieklasyczne algorytmy, lecz w oparciu o podejście klasyczne. Jest to precyzyjnie rzecz ujmując algorytm dla chipu typu Gapped z całkowicie wyłączoną weryfikacją. Spektrum wejściowe nie zawiera wtedy jakichkolwiek elementów nieklasycznych. W dalszej części rozdziału przedstawione zostaną także porównania wyników otrzymanych w oparciu o opracowane algorytmy z wynikami opublikowanymi w artykułach dotyczących innych algorytmów dla klasycznego lub też i nieklasycznego sekwencjonowania przez hybrydyzację.

Zanim przedstawione zostaną pierwsze wyniki eksperymentów, w następnym podrozdziale opisane zostaną parametry, które zastosowano dla poszczególnych algorytmów.

8.2 Parametry, zbiór testowy

Jak zostało już w pewnej mierze opisano wcześniej w rozdziałach poświęconych dokładnej konstrukcji zaproponowanych algorytmów, istnieje kilka podstawowych parametrów, według których symulowany jest eksperyment hybrydyzacyjny, jak i regulujących pracę samego algorytmu rekonstrukcji sekwencji DNA. Zostaną one teraz wymienione oraz opisane dokładniej.

1. k - parametr odpowiedzialny za długość oligonukleotydów dla omawianych chipów. Określa on też pojemność chipu, a z punktu widzenia algorytmu - długość elementów w dwóch częściach spektrum. W eksperymentach przyjmowane będą trzy wartości, dla których testowane były algorytmy dla chipów typu Gapped i Alternating: 8, 9 i 10. Odpowiednio dla tych wartości chipy zawierałyby wtedy 131072, 524288 oraz 2097152 sond (suma dwóch połówek chipów), chip klasyczny dla tych samych wartości k zawiera odpowiednio 65536, 262144 i 1048576 sond na powierzchni. Jeśli chodzi o chip typu Binary, parametr k w jego wypadku był równy odpowiednio 14, 16 oraz 18, co przekładało się na dłuższe oligonukleotydy, lecz ich całkowita liczba na chipie DNA była taka sama jak w przypadku odpowiednich wartości k dla chipów typu Gapped i Alternating. Dokładne zestawienie charakterystyk chipów dla parametru k przedstawia tabela 8.1.

TABLICA 8.1: Zestawienie podstawowych parametrów chipów używanych w obliczeniach, l_1 i l_2 - długości oligonukleotydów w częściach chipu, S_1 i S_2 - liczba sond z oligonukleotydami o danej długości, S - całkowita pojemność chipu

k	l_1	l_2	S_1	S_2	S
Chip klasyczny					
8	8	n.d.	65536	n.d.	65536
9	9	n.d.	262144	n.d.	262144
10	10	n.d.	1048576	n.d.	1048576
Gapped Chip					
8	8	15	65536	65536	131072
9	9	17	262144	262144	524288
10	10	19	1048576	1048576	2097152
Alternating Chip					
8	8	15	65536	65536	131072
9	9	17	262144	262144	524288
10	10	19	1048576	1048576	2097152
Binary Chip					
14	14	14	65536	65536	262144
16	16	16	262144	262144	524288
18	18	18	1048576	1048576	2097152

Jak widać z tabeli, dla każdego testowanego chipu istnieją jego trzy "wersje" różniące się od siebie wielkością, zależnie od wartości parametru k. Na potrzeby czytelności dalszych części tego rozdziału pracy przyjęto, że dla każdego chipu jego trzy podstawowe wersje oznaczane będą rzymskimi cyframi I, II oraz III. Chipy typu I to najmniejsze wersje (parametr k w kolejności występowania chipów w

tabeli 8.1 to 8, 8, 8 oraz 14), natomiast III typ oznacza największe chipy, z najwyższą liczbą sond, dla parametru k równego kolejno 10, 10, 10 oraz 18. O ile nie zostanie napisane inaczej, wyniki dla różnych następnych parametrów tutaj opisywanych będą przedstawiane w kolejności od najmniejszych do największych wersji chipów.

2. Obecność błędów hybrydyzacji w spektrum jest kolejnym parametrem rozpatrywanym w symulacji nieklasycznego sekwencjonowania DNA. Są możliwe cztery przypadki, w których testowano zaproponowane algorytmy. Pierwszy to brak jakichkolwiek błędów hybrydyzacji, następnie przypadek z błędami pozytywnymi, z błędami negatywnymi (ogólnego rodzaju) oraz ostatni przypadek, w którym w spektrum występują wszystkie rodzaje błędów.

3. Przybliżona liczba błędów w spektrum. Ten parametr dotyczy tylko przypadku, w którym występują oba rodzaje błędów hybrydyzacji. Niemożliwe jest wtedy precyzyjne określenie ich liczby, dlatego przyjmowana jest wartość dokładna (nierealistyczna) oraz wartości przybliżone. W przedstawionych wynikach pokazano rezultaty zarówno przy, jak już powiedziano - nierealistycznej wiedzy na temat liczby obu rodzajów błędów oddzielnie, oraz dla przybliżonych wartości 10%, 15% i 20% błędów w spektrum (realna liczba błędów będzie mniejsza).

4. Czas obliczeń w ogromnej większości eksperymentów wynosił 60 sekund. Jest to bardzo mały zakres czasu, zważywszy na ogrom przestrzeni rozwiązań przy długich sekwencjach DNA i niemożności określenia dokładnej liczby błędów. Wystarczał on jednak na przeprowadzenie testów pokazujących skuteczność zaproponowanych rozwiązań.

Jeśli chodzi o zbiór testowy, z bazy danych GenBank pobrano kilka bardzo długich sekwencji DNA (sekwencje kodujące), które połączone zostały w jeden, bardzo długi łańcuch DNA liczący około miliona par zasad. Z tej sekwencji przy każdej próbie losowana była podsekwencja o pewnej określonej długości, która następnie była pobierana jako łańcuch DNA, który musi być zrekonstruowany przez dany algorytm przy określonym spektrum pochodzącym z symulacji eksperymentu hybrydyzacyjnego. Sekwencje pobrane z GenBanku miały długość około kilkudziesięciu tysięcy par zasad każda. Wymienione są one poniżej, wraz z adresem internetowym, pod którym są dostępne:

1) $http://www.ncbi.nlm.nih.gov/nuccore/AC243774.1$ Homo sapiens FOSMID clone ABC27-711I14 from chromosome 1, complete sequence

2) $http://www.ncbi.nlm.nih.gov/nuccore/AC243632.2$ Homo sapiens chromosome 10 clone ABC12-46887700F22, WORKING DRAFT SEQUENCE

3) *http* : *//www.ncbi.nlm.nih.gov/nuccore/AC*243631.2 Homo sapiens chromosome 16 clone ABC10-44585000K16, WORKING DRAFT SEQUENCE

4) *http* : *//www.ncbi.nlm.nih.gov/nuccore/AC*243756.2 Homo sapiens chromosome UNK clone CH17-385C13, WORKING DRAFT SEQUENCE, 3 unordered pieces

5) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_028089.1 Homo sapiens protein phosphatase 1, regulatory (inhibitor) subunit 3B (PPP1R3B), RefSeqGene on chromosome 8

6) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_028088.1 Homo sapiens G-protein signaling modulator 1 (GPSM1), RefSeqGene on chromosome 9

7) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_028087.1 Homo sapiens toll-like receptor 6 (TLR6), RefSeqGene on chromosome 4

8) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_028084.1 Homo sapiens SH3-domain GRB2-like 1 (SH3GL1), RefSeqGene on chromosome 19

9) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_028083.1 Homo sapiens dynein, axonemal, light chain 1 (DNAL1), RefSeqGene on chromosome 14

10) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_016429.1 Homo sapiens prolyl endopeptidase like (PREPL), RefSeqGene on chromosome 2

11) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_008212 Homo sapiens von Hippel-Lindau tumor suppressor (VHL), RefSeqGene on chromosome 3

12) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_000013.3 Homo sapiens MHC class III complement gene cluster, monomodular haplotype (MCGC@) on chromosome 6

13) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_028086.1 Homo sapiens premelanosome protein (PMEL), RefSeqGene on chromosome 12

14) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_011630 Homo sapiens endothelin receptor type B (EDNRB), RefSeqGene on chromosome 13

15) *http* : *//www.ncbi.nlm.nih.gov/nuccore/FP*700187.20 Homo sapiens chromosome 18 clone CH17-221O5, WORKING DRAFT SEQUENCE, 16 unordered pieces

16) *http* : *//www.ncbi.nlm.nih.gov/nuccore/NG*_012089.1 Homo sapiens interleukin 10 receptor, beta (IL10RB), RefSeqGene on chromosome 21

Dane wejściowe algorytmów zostały już wcześniej opisane, jednak należy jeszcze doprecyzować, w jakiej formie algorytm uzyskuje spektrum DNA. Przed rozpoczęciem pracy algorytmu, odpowiednie procedury generują zbiór sond, które hybrydyzowałyby w idealnym przypadku z testowaną sekwencją DNA. Powstaje więc spektrum idealne. Z uwagi jednak na specyfikę takiego podejścia, podane sondy są ułożone w kolejności, w której mogą budować prawidłową sekwencję. Ponieważ powodowałoby to nieprawidłowość wyników (algorytm miałby bardzo ułatwione zadane, zaczynając zawsze od pierwszego wierzchołka niezależnie od chipu), muszą one zostać wymieszane, tj. ich kolejność musi być losowa, nie powiązana z budową badanego DNA. Zbiory obu połówek spektrum dla chipów nieklasycznych (lub jeden zbiór w przypadku chipu klasycznego) podlegają losowemu wymieszaniu w następujący sposób:

1) algorytm losuje dwie różne lokalizacje elementów w zbiorze, który aktualnie miesza;
2) jeśli wartości te są równe, lub choć jedna dotyczy elementu pierwszego, losowanie 1) jest powtarzane;
3) para wylosowanych elementów jest w zbiorze zamieniana miejscami;
4) kroki 1) - 3) są powtarzane tyle razy, ile wynosi trzykrotna liczba elementów zbioru.

Jeżeli w spektrum mają występować błędy negatywne, wymagana liczba elementów jest losowo usuwana z odpowiednich list. Następnie, jeśli jest to wymagane, do listy dodawane są losowe elementy, których liczba jest równa wymaganej liczbie błędów pozytywnych. Element jest traktowany jako błąd pozytywny tylko, jeżeli wcześniej w zbiorze oryginalnym nie występował, (tj. nie występował w zbiorze przed usunięciem elementów jako błędów negatywnych). Tak przygotowana lista mieszana jest po raz ostatni, zgodnie z podanym schematem. Podejście to zapewnia, że algorytm układając w czasie pracy listy kolejnych następników do odwiedzenia nie preferuje żadnych wierzchołków.

8.3 Brak błędów hybrydyzacji

Jako pierwszy zostanie przedstawiony przypadek bez błędów hybrydyzacji. Jak napisano już wcześniej, jest on dość nierealistyczny, ponieważ błędy hybrydyzacji są bardzo trudne do uniknięcia nawet przy zastosowaniu bardzo precyzyjnych technik w fazie biochemicznej SBH. Błędy negatywne wynikające z powtórzeń są praktycznie niemożliwe do wyeliminowania, ponieważ zależą one od budowy danego DNA. Teoretycznie można zmniejszać ich liczbę stosując dłuższe oligonukleotydy w procesie produkcji chipu. Ograniczeniem jest jednak pojemność chipu (a raczej związania z nią technologia wytwarzania), dlatego największe realistyczne chipy raczej nie przekraczają miliona sond, a i to w skrajnych przypadkach. Chip klasyczny służy głównie jako porównanie dla algorytmów nieklasycznych, które operując na danych nawet z mniejszych chipów mają lepszą lub choćby zbliżoną skuteczność jeśli chodzi o jednoznaczne sekwencjonowanie DNA.

Tabela 8.2 zawiera pierwsze wyniki eksperymentów dla różnych długości DNA przy zerowej liczbie błędów, dla najmniejszych badanych w pracu chipów każdego rodzaju.

Tablica 8.2: Liczba uruchomień algorytmów dla chipów I typu na 100, dla których algorytm zwrócił przynajmniej jedno rozwiązanie w czasie 60 sekund.

	Długość DNA						
Chip	100	200	300	400	500	600	700
Chip klasyczny	100	100	100	100	100	100	100
Gapped chip	100	100	100	100	100	100	100
Alternating chip	100	100	100	100	100	100	100
Binary chip	100	100	100	100	100	100	100

W znakomitej większości kolejne podrozdziały będą zawierać tabele o podobnej konstrukcji, gdzie zmiany dotyczyć będą rodzaju błędów występujących w spektrum czy innych parametrów określanych na potrzeby testu. Będzie to oczywiście wyraźnie opisane przed podaniem wyników w kolejnych tabelach. W tego rodzaju tabeli każda komórka zawiera liczbę uruchomień algorytmu dla różnych łańcuchów DNA (lecz o tej samej długości), dla których algorytm w zadanym czasie 1 minuty podał jakiekolwiek rozwiązanie. Powyższa tabela jest pod względem różnorodności wyników dość uboga, ponieważ nawet dla największych testowanych DNA algorytm w ciągu 60 sekund był w stanie podać przynajmniej jedno rozwiązanie dla każdej ze 100 różnych losowanych sekwencji DNA. Dla chipów II i III typu nie będą już podawana kolejne wyniki, ponieważ jak można się domyślić są one identyczne. Ponieważ maksymalna liczba prób dla danego DNA to 100, wyniki te równie dobrze można interpretować jako procentową szansę otrzymania rozwiązania w zadanym czasie.

Ciekawsze są dwie następne tabele - 8.3 oraz 8.4, prezentujące liczbę uruchomień, dla

których w 60 sekund algorytm był w stanie podać dokładnie jedno rozwiązanie oraz średnią liczbę rozwiązań w każdej ze 100 prób na różnych sekwencjach.

Tablica 8.3: Liczba uruchomień algorytmów dla chipów I typu, dla których algorytm zwrócił dokładnie jedno rozwiązanie w czasie 60 sekund.

	Długość DNA						
Chip	100	200	300	400	500	600	700
Chip klasyczny	96	77	39	21	2	0	0
Gapped chip	100	100	100	99	93	98	93
Alternating chip	100	100	100	99	100	100	100
Binary chip	100	100	100	100	100	100	100

Tablica 8.4: Średnia liczba rozwiązań algorytmów dla chipów I typu w czasie 60 sekund.

	Długość DNA						
Chip	100	200	300	400	500	600	700
Chip klasyczny	1.04	1.83	3.71	23.20	140	315	1331
Gapped chip	1.00	1.00	1.00	1.01	1.07	1.02	1.07
Alternating chip	1.00	1.00	1.00	1.01	1.00	1.00	1.00
Binary chip	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Wyraźnie widać, że dla chipu klasycznego wraz ze wzrostem długości badanej sekwencji DNA, drastycznie rośnie liczba rozwiązań niejednoznacznych. Chip typu Gapped, który od klasycznego różni się teoretycznie najmniej, ma jednak o wiele lepszą skuteczność w tym względzie dzięki weryfikacji kolejnych wierzchołków dodawanych do rozwiązania, jak zostało to już dokładnie opisane w Rozdziale 5. Chipy typu Alternating i Binary radzą sobie dość dobrze, będąc w stanie precyzyjnie podawać jedno rozwiązanie w zadanym czasie. Należy tutaj jednak zwrócić uwagę na niezmiernie ważną rzecz: ograniczenie czasowe. Nie jest powiedziane, że algorytmy dla tych chipów mając nieograniczony czas obliczeń nie byłyby w stanie podać więcej niż jednego rozwiązania dopuszczalnego. Problem w tego typu badaniach dotyczy wielkości przestrzeni rozwiązań. Dla małych instancji problemu (długości DNA) zostaną dalej w tym rozdziale przedstawione osobne wyniki dotyczące całkowitej liczby rozwiązań dopuszczalnych. Niestety dla DNA o długości 400 czy więcej par zasad czas obliczeń wzrasta do poziomu, który uniemożliwia efektywne przeprowadzenie wielu testów dla różnych fragmentów DNA. Dla najdłuższych badanych sekwencji czas obliczeń rośnie do dni lub nawet tygodni, w zależności od danej sekwencji DNA.

Należy również dodać, że wyniki prezentowane w tabeli 8.2 dla chipów nieklasycznych pokrywają się w całości z liczbą prób, w których odnaleziono *prawidłowe* DNA.

Dla chipu klasycznego tylko w przypadku DNA o długości 700 par zasad, w 94 przypadkach zbiór rozwiązań zawierał rozwiązanie prawidłowe. Dla krótszych sekwencji, zbiory te również w całości się pokrywały, tak jak dla chipów nieklasycznych.

Kolejne dwie tabele - 8.5 oraz 8.6 stanowią uzupełnienie poprzednich. Przedstawiają one ten sam rodzaj wyników, tym razem jednak dla największych chipów, czyli dla ich *III* typu.

Tablica 8.5: Liczba uruchomień algorytmów dla chipów *III* typu, dla których algorytm zwrócił dokładnie jedno rozwiązanie w czasie 60 sekund.

	Długość DNA						
Chip	100	200	300	400	500	600	700
Chip klasyczny	100	100	98	98	92	87	77
Gapped chip	100	100	100	100	100	100	100
Alternating chip	100	100	100	100	100	100	100
Binary chip	100	100	100	100	100	100	100

Tablica 8.6: Średnia liczba rozwiązań algorytmów dla chipów *III* typu w czasie 60 sekund.

	Długość DNA						
Chip	100	200	300	400	500	600	700
Chip klasyczny	1.00	1.00	1.02	1.02	1.15	1.20	1.56
Gapped chip	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Alternating chip	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Binary chip	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Jak widać z tabel 8.5 oraz 8.6, algorytmy dla chipów nieklasycznych podają precyzyjnie jedno rozwiązanie (w zadanym czasie), gdy już po minucie algorytm pracujący na klasycznym spektrum generuje rozwiązania niejednoznaczne. Dla wszystkich przypadków to jedno rozwiązanie było badanym DNA, w przypadku chipu klasycznego zbiór rozwiązań zawsze prawidłowe DNA zawierał.

8.4 Błędy pozytywne w spektrum DNA

Kolejna seria testów dotyczyła wpływu błędów pozytywnych w spektrum na jakość wyników zaproponowanych algorytmów nieklasycznych. Tabela 8.7 prezentuje wyniki dla najmniejszych badanych w pracy wersji chipów. Każda komórka tabeli zawiera liczbę uruchomień algorytmu dla różnych sekwencji DNA, w których algorytm w czasie 60 sekund podał tylko jedno rozwiązanie. W tym przypadku jak już napisano, algorytm wciąż zna dokładną liczbę elementów, które musi użyć ze spektrum (z maksymalnym możliwym dla chipu nałożeniem) równą liczbie elementów spektrum idealnego.

W tabeli widać dość wyraźnie, że algorytmy dla chipów nieklasycznych prezentują bardzo dużą odporność na występowanie błędów pozytywnych w spektrum. Zawarty w każdym algorytmie nieklasycznym mechanizm weryfikacji pozwala eliminować nieprawidłowe ścieżki w grafie już na poziomie ich tworzenia z poszczególnych wierzchołków. Oczywiście błędy pozytywne w podzbiorze spektrum odpowiedzialnym za weryfikację (Gapped i Alternating Chip) mogą powodować akceptację nieprawidłowych rozgałęzień, jest to jednak generalnie rzadko spotykana sytuacja, co tabela 8.7 zdaje się potwierdzać.

Wyniki dla algorytmów nieklasycznych, w warunkach użycia jeszcze większych chipów (II i III typu) nie staną się już generalnie lepsze, dlatego ostatnie dwie tabele, odpowiednio 8.8 i 8.9 prezentują wyniki tylko dla algorytmu operującego na chipie klasycznym III typu (ponad milion sond na powierzchni). Tabela 8.8 przedstawia liczbę uruchomień algorytmu (limit 100), w których w ciągu minuty algorytm podał tylko jedno rozwiązanie, a tabela 8.9 - średnią liczbę rozwiązań niejednoznacznych.

Na podstawie tabel tych widać wyraźnie, że przy sekwencjonowaniu DNA o długości powyżej 400 par zasad znacznie zwiększa się prawdopodobieństwo wystąpienia rozwiązań niejednoznacznych. Wciąż należy pamiętać, że mówimy tutaj o przypadkach ograniczenia czasowego do 60 sekund, w którym spora część przestrzeni rozwiązań wciąż nie została przeszukana (zwłaszcza w przypadku dłuższych fragmentów DNA). Wartości te więc (tak jak i średnie dla algorytmów nieklasycznych) mogą być wyższe, nie zmienia to jednak faktu, że wyraźnie widać tutaj przewagę rozwiązań nieklasycznych.

Tablica 8.7: Liczba uruchomień algorytmów dla chipów I typu, dla których algorytm zwrócił dokładnie jedno rozwiązanie w czasie 60 sekund (błędy pozytywne)

	Procent błędów pozytywnych				
Długość DNA	1%	2%	3%	4%	5%
Chip klasyczny					
100	98	99	99	99	100
200	80	85	79	84	83
300	45	47	54	50	46
400	19	19	21	18	19
500	6	2	4	3	3
600	0	0	0	0	0
700	0	0	0	0	0
Gapped Chip					
100	100	100	100	100	100
200	100	100	100	100	100
300	100	100	100	100	99
400	99	100	99	100	100
500	99	100	100	99	100
600	99	98	100	99	100
700	98	100	98	98	95
Alternating Chip					
100	100	100	100	100	100
200	100	100	100	100	100
300	99	100	100	100	100
400	100	100	100	100	100
500	100	100	100	100	100
600	100	100	100	100	99
700	100	100	100	100	100
Binary Chip					
100	100	100	100	100	100
200	100	100	100	100	100
300	100	100	100	100	100
400	100	100	100	100	100
500	100	100	100	100	100
600	100	100	100	100	100
700	100	100	100	100	100

Tablica 8.8: Liczba uruchomień algorytmu klasycznego (chip *III* typu), dla których algorytm zwrócił dokładnie jedno rozwiązanie w czasie 60 sekund (błędy pozytywne)

	Procent błędów pozytywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100	99	100	100	99
200	99	98	100	100	99
300	98	99	96	98	99
400	96	97	96	93	97
500	89	90	90	95	92
600	88	84	93	80	88
700	75	75	70	78	76

Tablica 8.9: Średnia liczba rozwiązań dla algorytmu klasycznego chipu *III* typu (błędy pozytywne)

	Procent błędów pozytywnych				
Długość DNA	1%	2%	3%	4%	5%
100	1.00	1.01	1.00	1.00	1.01
200	1.01	1.02	1.00	1.00	1.01
300	1.03	1.03	1.04	1.02	1.01
400	1.04	1.06	1.09	1.11	1.03
500	1.14	1.12	1.13	1.09	1.12
600	1.18	1.27	1.09	1.60	1.31
700	1.46	1.40	1.46	1.54	1.40

8.5 Błędy negatywne w spektrum DNA

Błędy negatywne w spektrum powodują, że niemożliwe staje się nałożenie kolejnych elementów ścieżki na siebie na maksymalnej dozwolonej długości. W efekcie muszą zostać użyte połączenia między parami wierzchołków, charakteryzujące się mniejszym nałożeniem niż maksymalne. Problemem jest oczywiście określenie kiedy takie nałożenie faktycznie powinno nastąpić oraz to, jaka wartość nałożenia (jeżeli nie maksymalna) ma zostać wybrana w przypadku danego wierzchołka. To co daje się precyzyjnie określić to limit, powyżej którego algorytm przestaje używać połączeń między wierzchołkami charakteryzującymi się mniejszymi nałożeniami. Wynika on z różnicy liczby elementów spektrum idealnego i danego. Błędy negatywne wpływają już dość znacznie na jakość rozwiązań, co wykażą dalsze zestawienia wyników testów wszystkich badanych algorytmów. W tabelach 8.10, 8.11, 8.12, 8.13 zaprezentowane zostały wyniki dla chipów *I* typu. W każdym polu tabeli, w zależności od długości DNA i procentu błędów negatywnych,

wartości liczbowe określają w ilu próbach na sto algorytm w ciągu 60 sekund zwrócił jakiekolwiek rozwiązanie oraz ile razy w ciągu minuty było to tylko jedno rozwiązanie (wartość w nawiasie).

Tablica 8.10: Liczba uruchomień dla chipu klasycznego *I* typu, dla których algorytm zwrócił jedno i wiele rozwiązań w czasie 60 sekund (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100(98)	98(93)	99(93)	97(95)	95(92)
200	98(59)	98(69)	87(69)	74(58)	63(52)
300	82(35)	56(27)	35(27)	40(32)	28(22)
400	58(16)	25(12)	18(12)	10(9)	15(9)
500	35(17)	21(4)	8(4)	11(10)	6(5)
600	21(9)	3(5)	8(5)	4(2)	3(0)
700	9(5)	4(1)	2(1)	2(0)	1(1)

Tablica 8.11: Liczba uruchomień dla chipu typu Gapped *I* typu, dla których algorytm zwrócił jedno i wiele rozwiązań w czasie 60 sekund (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100(100)	99(94)	96(94)	94(93)	87(86)
200	95(91)	93(80)	82(80)	78(76)	72(70)
300	91(86)	79(64)	66(64)	64(56)	61(57)
400	77(71)	69(52)	58(52)	40(32)	46(37)
500	72(61)	55(32)	46(32)	39(24)	30(20)
600	56(44)	44(25)	31(25)	29(14)	23(15)
700	43(30)	36(15)	22(15)	24(11)	12(9)

Wyraźnie widoczna jest różnica skuteczności chipów nieklasycznych dla dłuższych fragmentów DNA, w szczególności jeśli chodzi o chip typu Binary (tabela 8.13). Ograniczenie czasowe nie pozwala precyzyjnie określić ani nawet przybliżyć, ile rozwiązań niejednoznacznych algorytm podałby, mając wystarczającą ilość czasu na przeszukanie całej przestrzeni rozwiązań. Tabela 8.14 pokazuje średnią liczbę rozwiązań dla danego spektrum i danych parametrów testu. Widać ogólną tendencję, że w tym samym zakresie czasu w stu próbach dla każdej badanej długości DNA i dla danego procentu błędów negatywnych, algorytmy nieklasyczne zwracają mniejszą liczbę rozwiązań niejednoznacznych, niż ich klasyczny odpowiednik.

Tablica 8.12: Liczba uruchomień dla chipu typu Alternating I typu, dla których algorytm zwrócił jedno i wiele rozwiązań w czasie 60 sekund (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100(100)	97(94)	94(94)	94(94)	88(85)
200	98(98)	92(83)	86(83)	89(88)	74(70)
300	91(91)	90(77)	80(77)	72(66)	52(49)
400	78(77)	80(62)	65(62)	43(41)	35(30)
500	72(69)	61(24)	26(24)	14(12)	6(6)
600	65(62)	36(9)	9(9)	7(7)	4(4)
700	48(45)	17(7)	10(7)	2(2)	2(1)

Tablica 8.13: Liczba uruchomień dla chipu typu Binary I typu, dla których algorytm zwrócił jedno i wiele rozwiązań w czasie 60 sekund (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100(99)	97(97)	97(97)	91(91)	86(86)
200	99(99)	96(86)	86(86)	94(94)	84(84)
300	96(95)	92(86)	87(86)	82(82)	80(80)
400	94(94)	90(82)	83(82)	76(76)	69(69)
500	90(89)	76(71)	72(71)	61(60)	45(43)
600	85(85)	69(59)	60(59)	55(51)	29(27)
700	76(76)	58(42)	43(42)	30(30)	20(19)

Z tabeli 8.14 oraz czterech tabel poprzednich (8.10, 8.11, 8.12, 8.13) widać, że wraz ze wzrostem liczby błędów negatywnych, oraz zwiększaniem się długości sekwencji DNA, rośnie prawdopodobieństwo wystąpienia rozwiązań niejednoznacznych. Jest ono jednak zasadniczo mniejsze w przypadku algorytmów operujących na nieklasycznych spektrach, jak zostało już teoretycznie przewidziane w sekcjach dotyczących opisu odpowiednich chipów, oraz w opisie mechanizmów weryfikacji zawartych w samych algorytmach. Na osobną uwagę zasługuje ostatni wiersz sekcji tabeli dotyczącej chipu klasycznego. Ma on dość niezwykłe wyniki, z których mogłoby wynikać, że przy bardzo dużych sekwencjach liczba rozwiązań niejednoznacznych maleje, lub przynajmniej przestaje narastać. Byłaby to jednak nieprawidłowa interpretacja. Powody tego typu odchyleń wyjaśnia tabela 8.10, w której przedstawiono liczbę prób zakończonych podaniem przez algorytm jakiegokolwiek rozwiązania oraz (wartość w nawiasie) liczba prób, w których w ciągu minuty algorytm podał tylko jedno rozwiązanie. Różnica między tymi wartościami to faktyczna liczba instancji, dla których algorytm zwrócił więcej niż jedno rozwiązanie.

Tablica 8.14: Średnia liczba rozwiązań niejednoznacznych dla chipów I typu (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
Chip klasyczny					
100	1.02	1.10	1.10	1.02	1.07
200	1.94	1.50	1.33	1.39	1.25
300	3.41	2.00	1.45	1.32	1.28
400	5.67	1.36	1.38	1.10	1.73
500	2.71	1.57	2.87	1.18	6.83
600	5.09	1.66	2.75	2.25	18.00
700	2.22	1.00	1.50	14.00	1.00
Gapped Chip					
100	1.00	1.01	1.02	1.01	1.02
200	1.04	1.04	1.02	1.06	1.30
300	1.06	1.10	1.03	1.17	1.06
400	1.12	1.47	1.17	11.4	1.30
500	1.23	1.14	1.69	2.53	1.56
600	1.23	1.38	1.25	5.65	2.08
700	1.41	4.36	1.77	2.25	1.58
Alternating Chip					
100	1.00	1.00	1.00	1.00	1.05
200	1.00	1.02	1.03	1.01	1.10
300	1.00	1.04	1.10	1.11	1.05
400	1.01	1.06	1.06	1.13	1.25
500	1.04	1.03	1.11	1.21	1.00
600	1.06	1.02	1.00	1.00	1.00
700	1.06	1.05	2.00	1.00	1.50
Binary Chip					
100	1.01	1.02	1.00	1.00	1.00
200	1.00	1.00	1.00	1.00	1.00
300	1.01	1.00	1.02	1.00	1.00
400	1.00	1.00	1.01	1.00	1.00
500	1.01	1.01	1.01	1.01	1.04
600	1.00	1.00	1.01	1.07	1.13
700	1.00	1.01	1.09	1.00	1.05

Jeśli ta różnica jest niewielka, wpływ na wartość średniej jest odpowiednio mniejszy. Widać, że liczba prób zakończonych wieloma rozwiązaniami maleje wraz ze wzrostem długości DNA i liczby błędów negatywnych. Drugim powodem, który odpowiada za opisany właśnie fakt oraz za niezwykłość omawianego wiersza tabeli 8.14 jest to, że wraz ze wzrostem tych wartości, rośnie przestrzeń rozwiązań, którą algorytm musi przeszukać. Oznacza to, że maleje prawdopodobieństwo podania jakiegokolwiek rozwiązania w tak krótkim czasie jak 60 sekund, a jeśli już to następuje, trudno takie instancje z uwagi na ich niewielką ilość w zbiorze prób uznać za reprezentacyjne. Dlatego jest dość oczywistym, że w przypadku zwiększenia czasu obliczeń liczba rozwiązań niejednoznacznych wzrośnie.

Jak już w niniejszym rozdziale wspominano, 60 sekund na jedną próbę algorytmu to czas wyjątkowo krótki. Z drugiej jednak strony dla instancji z długim łańcuchem DNA i dużą ilością błędów negatywnych, czas na przeszukanie przestrzeni rozwiązań jest trudny do określenia. W dużym przybliżeniu jednak mierzony nawet nie w godzinach lub dniach, lecz wręcz w tygodniach ciągłego przeszukiwania. Ograniczenie tego czasu do niewielkiego ułamka ma oczywisty wpływ na jakość prezentowanych tutaj wyników. Rekompensowane jest to liczbą prób, co pozwala obserwować ogólne trendy i zasady zachowania się algorytmów z punktu widzenia ich efektywności, w zależności od danych wejściowych. Badania dotyczące przypadków, w których czas obliczeń nie był tak drastycznie ograniczany zostaną przedstawione dalej w pracy.

Należy tu zaznaczyć pewną istotną rzecz związaną z wcześniej zdefiniowanym i opisanym tzw. prawdopodobieństwem rozgałęzień. Wzory i wyliczenia przytoczone z artykułu [40] w odpowiednich podrozdziałach dotyczących zaproponowanych algorytmów nieklasycznych nie dotyczą przypadków, w których występują błędy hybrydyzacji. Mają one wartość teoretyczną, pozwalającą oceniać praktyczną skuteczność zaproponowanych w artykule chipów nieklasycznych w stosunku do podejścia klasycznego. Nie określają one jednak nawet przybliżonych wartości efektywności chipów dla przypadków z różnymi rodzajami błędów i z różnym procentem ich występowania w stosunku do całego spektrum DNA. Rozsądne jest przyjęcie założenia, że przy tych samych wartościach określających długość DNA i liczbę błędów, chipy nieklasyczne będą radzić sobie lepiej niż ich klasyczny odpowiednik. Nie ma jednak teoretycznych obliczeń dotyczących tejże skuteczności. Możemy o niej mówić na podstawie prezentowanych w niniejszej pracy wyników testów.

Tabele 8.15, 8.16, 8.17 oraz 8.18 przedstawiają wyniki testów dla chipów *II* typu. Jak poprzednio pierwsza wartość pokazuje liczbę prób, w których w 60 sekund algorytm wygenerował przynajmniej jedno rozwiązanie, w nawiasie natomiast podana jest liczba prób, w których było to tylko jedno rozwiązanie.

Tablica 8.15: Liczba uruchomień na 100, dla chipu klasycznego *II* typu, dla których algorytm zwrócił jedno i wiele rozwiązań w czasie 60 sekund (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100(98)	96(94)	96(94)	96(94)	98(97)
200	100(94)	97(83)	86(83)	80(79)	76(72)
300	93(78)	81(61)	66(61)	52(49)	52(47)
400	78(60)	61(38)	39(38)	40(38)	28(28)
500	53(36)	26(26)	26(26)	21(18)	21(20)
600	36(25)	19(12)	14(12)	12(12)	12(11)
700	24(16)	11(11)	12(11)	5(5)	5(5)

Tablica 8.16: Liczba uruchomień na 100, dla chipu typu Gapped *II* typu, dla których algorytm zwrócił jedno i wiele rozwiązań w czasie 60 sekund (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100(99)	99(93)	95(93)	95(94)	92(91)
200	99(97)	93(84)	84(84)	84(84)	83(83)
300	94(93)	86(77)	77(77)	73(72)	80(78)
400	93(91)	73(69)	70(69)	67(64)	63(60)
500	81(81)	80(61)	61(61)	59(57)	51(49)
600	70(69)	55(48)	52(48)	62(56)	52(46)
700	71(68)	50(39)	44(39)	48(46)	38(36)

Tablica 8.17: Liczba uruchomień na 100, dla chipu typu Alternating *II* typu, dla których algorytm zwrócił jedno i wiele rozwiązań w czasie 60 sekund (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100(100)	99(97)	97(97)	94(93)	90(89)
200	99(99)	94(89)	89(89)	92(89)	85(83)
300	92(90)	85(85)	85(85)	86(84)	74(72)
400	89(89)	87(84)	85(84)	63(60)	56(52)
500	85(84)	72(69)	70(69)	54(53)	45(45)
600	78(78)	61(44)	48(44)	28(28)	27(24)
700	70(70)	57(35)	35(35)	32(30)	19(19)

Tablica 8.18: Liczba uruchomień na 100, dla chipu typu Binary *II* typu, dla których algorytm zwrócił jedno i wiele rozwiązań w czasie 60 sekund (błędy negatywne)

	Procent błędów negatywnych				
Długość DNA	1%	2%	3%	4%	5%
100	100(100)	95(95)	96(95)	96(96)	85(84)
200	99(99)	93(93)	93(93)	82(81)	88(87)
300	99(99)	95(92)	93(92)	87(87)	79(78)
400	93(91)	95(85)	85(85)	83(82)	78(78)
500	92(92)	80(74)	75(74)	77(76)	77(77)
600	87(86)	81(76)	77(76)	68(67)	66(65)
700	82(82)	75(77)	77(77)	76(74)	61(60)

Tutaj również widać przewagę algorytmów nieklasycznych, a szczególnie widoczna jest ona w przypadku chipu typu Binary.

8.6 Wszystkie rodzaje błędów w spektrum DNA

W tym podrozdziale przedstawione zostaną testy, w ramach których w spektrach DNA znajdują się oba rodzaje błędów hybrydyzacji. Jest to najtrudniejszy przypadek dla algorytmu, niestety najbardziej realistyczny z punktu widzenia metody sekwencjonowania przez hybrydyzację. W poprzednich przypadkach, kiedy błędy negatywne i pozytywne nie występują razem, algorytm mógł precyzyjnie obliczyć limit wierzchołków, które jeszcze należy odwiedzić poprzez nałożenie mniejsze niż maksymalne tak, aby zrekonstruować DNA o danej długości. W przypadku występowania tylko błędów pozytywnych należy użyć wierzchołków w liczbie równej liczności spektrum idealnego, przy czym tylko wierzchołki z maksymalnym nałożeniem są brane pod uwagę. W przypadku błędów negatywnych znany jest maksymalny limit użycia nałożeń mniejszych niż maksymalne, aby otrzymać DNA o odpowiedniej długości. Opisywany w tym podrozdziale przypadek obu rodzajów błędów w spektrum jest trudny także pod tym względem, że nie można precyzyjnie określić liczby wierzchołków, które muszą być odwiedzone. Wartość ta musi być określona w przybliżeniu. Bez niej przestrzeń rozwiązań zwiększa się, ponieważ algorytm testuje wszystkie połączenia wychodzące z każdego wierzchołka, niezależnie od tego ile razy użył już nałożeń mniejszych niż maksymalne (czyli innymi słowy ile błędów negatywnych już do tej pory "naprawił" w rekonstruowanej sekwencji / ścieżce). Testowanie wszystkich takich połączeń byłoby więc skrajnie nieefektywne z punktu widzenia czasu obliczeń.

8.6.1 Skuteczność algorytmów dla obu rodzajów błędów oraz porównanie z przypadkiem błędów negatywnych

W niniejszym podrozdziale przedstawione zostaną tabele wyników, w których zaprezentowane zostaną trzy wartości reprezentujące hipotetyczny limit błędów negatywnych przyjętych przez algorytm. Wartości te to 10%, 15% oraz 20% błędów w stosunku do liczby elementów spektrum idealnego. Są to wartości określające maksymalny limit użycia nałożeń mniejszych niż maksymalne dla dwóch wierzchołków. Rzeczywista liczba błędów w danych przypadkach pozostaje taka sama jak w poprzednich sekcjach, zmieniając się od 1% do 5% liczby elementów spektrum idealnego. Taka liczba elementów jest losowo zabierana ze spektrum, taka też liczba nieprawidłowych elementów (błędów pozytywnych) jest dodawana do spektrum zanim dany algorytm rozpocznie proces poszukiwania DNA. Limit, o którym mowa w tym podrozdziale, ma też bezpośredni wpływ na minimalną liczbę wierzchołków, która musi zostać użyta, aby algorytm zaakceptował rozwiązanie jako dopuszczalne. Tak na przykład dla limitu równego 10% algorytm akceptuje te rekonstrukcje DNA, do których budowy użył od 90% (do maksymalnie 100%) liczby elementów spektrum idealnego. Innymi słowy limit ten reprezentuje hipotetyczną maksymalną liczbę błędów negatywnych, którą może lecz nie musi zawierać spektrum.

Dość interesującym wydaje się porównanie wyników dla przypadków z wszystkimi rodzajami błędów w spektrum, przy różnych limitach ograniczających pracę algorytmów, z przypadkiem, kiedy występują tylko błędy negatywne opisane w poprzedniej sekcji. W przedstawionych tutaj tabelach podane zostaną zbiorcze wyniki dla algorytmu chipu klasycznego oraz zaproponowanych algorytmów dla chipów nieklasycznych.

Trzy sekcje tabeli odpowiadają zbiorczym wynikom dla tych samych długości DNA. Każda z nich jest podzielona na cztery wiersze, gdzie pierwszy zawsze dotyczy wyników dla przypadku, w którym w spektrum są tylko i wyłącznie błędy negatywne. Kolejne trzy to opisane już przypadki, kiedy oba rodzaje błędów występują jednocześnie. Przyjęto dla nich trzy opisane już limity błędów, które przyjmuje algorytm przed rozpoczęciem rekonstrukcji. Każda komórka zawiera trzy liczby:

1) pierwsza wartość to liczba przypadków na 100, w których algorytm przed upływem 60 sekund podał jakiekolwiek rozwiązanie;

2) wartość druga w zwykłym nawiasie to opisywana już wcześniej liczba przypadków, w których w ciągu 60 sekund algorytm podał tylko jedno rozwiązanie;

3) trzecia wartość podana w nawiasie kwadratowym to liczba przypadków, w których w zbiorze rozwiązań po 60 sekundach algorytm podał rozwiązanie będące prawidłową sekwencją DNA.

Trzecia wartość jest możliwa do ustalenia, ponieważ w przeciwieństwie do normalnego laboratoryjnego sekwencjonowania DNA, dysponujemy wiedzą o oryginalnej sekwencji, którą algorytm ma odtworzyć. Dzięki temu możliwe jest ustalenie czy sekwencja lub zbiór sekwencji, które podał algorytm po upływie ustalonego czasu, jest identyczna z badanym w danym przypadku DNA (lub czy zbiór rozwiązań je zawiera). Wartość ta jest szczególnie interesująca w kontekście bardzo ograniczonego czasu obliczeń. Pomimo przeszukania w niektórych wypadkach niewielkiego ułamka dostępnej przestrzeni rozwiązań, algorytm bardzo często natrafia na prawidłowe rozwiązanie. Druga tabela w zestawieniach prezentować będzie zbiorcze wartości średniej liczby rozwiązań.

8.6.1.1 Chip klasyczny

W tabelach 8.19, 8.20, 8.21 oraz 8.22 przedstawiono wyniki testów algorytmu dla DNA o długościach 300, 500 oraz 700 nukleotydów. Tabela 8.19 prezentuje wyniki dla chipu klasycznego I typu. Poza nawiasem podano liczbę prób na 100, w których algorytm w 60 sekund znalazł dowolny łańcuch DNA, w nawiasie zwykłym liczbę prób z tylko jednym rozwiązaniem w zadanym czasie, w nawiasie kwadratowym - liczbę prób, w których w zbiorze rozwiązań dopuszczalnych znalazło się prawidłowe badane DNA. Tabela 8.20 prezentuje średnią liczbę rozwiązań znalezionych w ciągu minuty.

Jak widać z tabel 8.19 i 8.20 najmniejszy z rozpatrywanych tutaj chipów klasycznych nie nadaje się do sekwencjonowania długich fragmentów DNA, zakładając obecność błędów hybrydyzacji. Szansa na jednoznaczne rozwiązanie jest w zasadzie zerowa, a liczba rozwiązań niejednoznacznych już w ciągu minuty osiąga wartość kilkudziesięciu sekwencji. Dlatego też identyczne testy przeprowadzono dla chipu III typu. W tabelach 8.21 i 8.22 zaprezentowano wyniki dla największego chipu klasycznego, analogicznie do już przedstawionych w dwóch poprzednich tabelach.

Ciekawe jest porównanie skuteczności algorytmu dla przypadku tylko z błędami negatywnymi (pierwszy wiersz każdej z trzech sekcji) do pozostałych, w których algorytm nie wie ile błędów rzeczywiście występuję. Algorytm dla błędów negatywnych w ciągu minuty ma mniejszą szansę podania jakiegokolwiek rozwiązania, jednak ich jakość jest zdecydowanie lepsza. Prawdopodobieństwo wystąpienia rozwiązań niejednoznacznych jest o wiele niższe, poza tym nawet w tak ograniczonym czasie istnieje szansa na odszukanie prawidłowej sekwencji.

Tablica 8.19: Chip klasyczny I typu, zestawienie zbiorcze wyników prób z jednym, wieloma oraz prawidłowym rozwiązaniem

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	82(35)[71]	56(34)[38]	35(27)[13]	40(32)[21]	28(22)[14]
300 (10%)	100(1)[11]	100(0)[7]	100(2)[9]	100(0)[5]	98(1)[8]
300 (15%)	100(1)[11]	100(0)[7]	100(2)[9]	100(0)[5]	100(1)[8]
300 (20%)	100(1)[7]	100(2)[16]	100(2)[13]	100(0)[4]	100(1)[12]
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	35(17)[4]	21(14)[1]	8(4)[0]	11(10)[0]	6(5)[0]
500 (10%)	100(0)[0]	100(1)[0]	100(0)[1]	100(0)[0]	98(0)[0]
500 (15%)	100(0)[0]	100(1)[0]	100(0)[1]	100(0)[0]	100(0)[0]
500 (20%)	100(0)[0]	100(1)[0]	100(0)[0]	100(1)[0]	100(0)[0]
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	9(5)[0]	4(4)[0]	2(1)[0]	2(0)[0]	1(1)[0]
700 (10%)	100(0)[0]	100(0)[0]	100(0)[0]	100(0)[0]	99(0)[0]
700 (15%)	100(0)[0]	100(0)[0]	100(0)[0]	100(0)[0]	100(0)[0]
700 (20%)	100(0)[0]	100(0)[0]	100(0)[0]	100(0)[0]	100(0)[0]

Tablica 8.20: Chip klasyczny I typu, zestawienie zbiorcze średniej liczby rozwiązań dla różnych wartości limitu błędów

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	3.4	2.0	1.4	1.3	1.3
300 (10%)	27.8	28.2	33.0	38.2	33.6
300 (15%)	25.3	31.1	34.7	50.4	53.9
300 (20%)	23.9	31.5	24.6	44.2	44.4
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	2.7	1.6	2.9	1.2	6.8
500 (10%)	29.3	47.1	42.9	54.7	69.6
500 (15%)	57.8	39.0	50.4	58.8	74.8
500 (20%)	26.0	45.0	47.0	62.8	66.7
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	2.2	1.0	1.5	14.0	1.0
700 (10%)	32.7	41.1	76.3	57.9	85.6
700 (15%)	34.1	45.4	74.6	80.7	94.2
700 (20%)	24.9	54.0	66.8	88.6	91.0

Tablica 8.21: Chip klasyczny *III* typu, zestawienie zbiorcze wyników prób z jednym, wieloma oraz prawidłowym rozwiązaniem

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	95(91)[95]	85(80)[82]	70(68)[69]	72(72)[71]	71(71)[70]
300 (10%)	88(65)[67]	78(63)[60]	84(64)[65]	84(58)[60]	78(58)[55]
300 (15%)	93(64)[75]	81(53)[59]	86(67)[71]	89(58)[64]	90(68)[66]
300 (20%)	91(66)[78]	80(54)[56]	85(66)[61]	78(57)[60]	87(68)[64]
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	63(57)[59]	52(51)[49]	43(40)[41]	47(46)[45]	40(40)[39]
500 (10%)	82(67)[41]	78(52)[38]	68(52)[37]	65(46)[27]	73(49)[43]
500 (15%)	79(49)[42]	73(57)[44]	80(56)[34]	64(51)[34]	69(51)[34]
500 (20%)	81(63)[42]	67(45)[41]	70(56)[36]	72(51)[35]	68(53)[31]
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	42(38)[33]	27(26)[21]	24(24)[19]	24(23)[20]	20(20)[18]
700 (10%)	76(56)[21]	76(59)[21]	63(43)[24]	52(36)[15]	59(37)[17]
700 (15%)	72(53)[25]	67(46)[25]	60(44)[21]	63(46)[16]	66(52)[19]
700 (20%)	74(55)[22]	63(45)[24]	58(43)[19]	62(47)[16]	55(38)[17]

Tablica 8.22: Chip klasyczny *III* typu, zestawienie zbiorcze średniej liczby rozwiązań dla różnych wartości limitu błędów

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	1.05	1.05	1.020	1.00	1.00
300 (10%)	2.57	1.62	1.71	2.98	1.65
300 (15%)	1.60	2.12	1.51	2.32	1.63
300 (20%)	1.43	3.80	1.36	1.67	1.63
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	1.15	1.010	2.09	1.02	1.00
500 (10%)	1.21	1.88	1.63	1.92	4.24
500 (15%)	2.24	1.34	1.71	1.92	1.55
500 (20%)	1.96	2.17	2.28	1.55	1.97
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	1.09	1.03	1.00	1.04	1.00
700 (10%)	1.48	1.38	3.36	1.38	3.35
700 (15%)	1.69	1.91	2.50	2.38	2.09
700 (20%)	1.89	2.53	3.63	1.72	1.70

8.6.1.2 Gapped Chip

W tabeli 8.23 przedstawione zostało podobne zestawienie dla chipu typu Gapped, tak jak poprzednio dla algorytmu chipu klasycznego. To co rzuca się w oczy to porównanie jakości rozwiązań w przypadku tylko z błędami negatywnymi z pozostałymi, gdzie przyjęto hipotetyczny limit błędów z braku możliwości jego precyzyjnego ustalenia. Ponieważ algorytm dla błędów negatywnych musi znaleźć rozwiązanie zawierające wszystkie elementy spektrum, nie powinno dziwić, że znajduje on w tym samym czasie mniej rozwiązań co w innych przedstawionych przypadkach, gdzie dopuszczalność znalezionych rekonstrukcji jest o wiele mniej restrykcyjna. Kolejną różnicą w stosunku do przypadków z obydwoma rodzajami błędów jest stosunek liczby prób z rozwiązaniem, do liczby prób z rozwiązaniem jednoznacznym. Widać że wartości te są do siebie o wiele bardziej zbliżone właśnie w przypadku występowania tylko i wyłącznie błędów negatywnych. Dość interesująco prezentuje się także trzecia wartość - liczba prób, w których w zbiorze rozwiązań znajdowało się badane DNA. Na przykład dla DNA o długości 700 par zasad i 1% błędów w spektrum, w przypadku tylko błędów negatywnych na 43 próby z rozwiązaniem, 30 zakończyło się podaniem rozwiązań jednoznacznych (w zadanym czasie), lecz w 41 na 43 w zbiorze rozwiązań występowało oryginalne DNA. Już dla przypadku z najniższym limitem 10%, kiedy w spektrum obecne są oba rodzaje błędów hybrydyzacji, algorytm w ciągu minuty podał rozwiązania w co prawda aż 81 przypadkach, lecz tylko w 33 były to rozwiązania jednoznaczne, a na 81 prób tylko w 37 w zbiorze rozwiązań znalazło się badane DNA. Oczywiście rozwiązania niejednoznaczne utrudniają szybkie sekwencjonowania metodą SBH, lecz interesujący jest fakt, że dla algorytmów i chipów nieklasycznych w wielu przypadkach bardzo szybko możliwe jest uzyskanie prawidłowej sekwencji DNA, jako rozwiązania jednoznacznego w zadanym czasie, ale także w zbiorze wielu rozwiązań dopuszczalnych. Oczywiście najlepszy stosunek tychże trzech podanych w każdej komórce tabeli wartości występuje w przypadku tylko z błędami negatywnymi. Druga z tabel (8.24) przedstawia średnią liczbę rozwiązań dla algorytmu chipu Gapped I typu.

Z tabeli 8.25 dość wyraźnie widać, że różnica między próbami z jakimkolwiek rozwiązaniem a rozwiązaniami niejednoznacznymi jest bardzo niewielka, przynajmniej w czasie pierwszych 60 sekund. Wartości średniej liczby rozwiązań (tabela 8.26) dla chipu typu Gapped w omawianym teście oscylują między 1.00 a 1.50, przy czym wpływ limitu na ich wielkość jest raczej trudny do ustalenia z uwagi na znaczny wpływ ograniczenia czasowego. Dla 60 sekund powoduje on, że algorytm nie ma czasu na wyszukanie wielu rozwiązań niejednoznacznych (jeśli w ogóle jakieś podaje w zadanym czasie). Porównując z analogiczną tabelą dla chipu klasycznego (tabela 8.22) widać dość wyraźnie, że w tym samym czasie liczba prób z rozwiązaniami niejednoznacznymi w stosunku do wszystkich

Tablica 8.23: Chip typu Gapped I typu, zestawienie zbiorcze wyników prób z jednym, wieloma oraz prawidłowym rozwiązaniem

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	91(86)[91]	79(72)[79]	66(64)[66]	64(56)[63]	61(57)[59]
300 (10%)	91(58)[76]	82(54)[75]	73(37)[59]	67(33)[62]	58(44)[54]
300 (15%)	98(57)[88]	84(41)[69]	81(35)[57]	72(46)[57]	74(44)[54]
300 (20%)	100(47)[82]	91(51)[78]	84(48)[64]	80(45)[51]	77(39)[52]
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	72(61)[71]	55(48)[52]	46(32)[41]	39(24)[37]	30(20)[28]
500 (10%)	87(54)[55]	71(39)[46]	61(33)[39]	48(29)[34]	29(16)[24]
500 (15%)	82(46)[55]	75(32)[39]	72(32)[36]	75(37)[29]	55(29)[25]
500 (20%)	95(56)[69]	80(44)[44]	79(28)[31]	78(33)[35]	71(30)[20]
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	43(30)[41]	36(21)[36]	22(15)[16]	24(11)[14]	12(9)[6]
700 (10%)	81(33)[37]	61(30)[26]	40(14)[14]	40(16)[17]	10(5)[6]
700 (15%)	82(44)[40]	85(38)[20]	84(29)[21]	59(24)[14]	48(21)[11]
700 (20%)	91(42)[37]	83(31)[25]	87(31)[27]	82(23)[16]	73(16)[8]

Tablica 8.24: Chip typu Gapped I typu, zestawienie zbiorcze średniej liczby rozwiązań dla różnych wartości limitu błędów

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	1.06	1.10	1.03	1.17	1.06
300 (10%)	1.63	1.52	1.93	2.00	4.58
300 (15%)	4.55	3.21	3.44	2.01	1.74
300 (20%)	2.26	2.06	3.26	10.6	6.02
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	1.23	1.14	1.69	2.53	1.56
500 (10%)	5.03	2.36	2.81	2.85	2.62
500 (15%)	1.98	5.64	4.97	3.96	6.32
500 (20%)	3.30	4.33	4.30	6.66	7.19
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	1.41	4.36	1.77	2.25	1.58
700 (10%)	3.74	2.09	2.97	3.60	1.70
700 (15%)	2.32	4.04	4.57	4.06	4.39
700 (20%)	3.18	5.50	8.21	6.95	12.00

prób zakończonych znalezieniem choć jednego DNA jest jednak znacznie mniejsza dla chipu Gapped. Innymi słowy w ciągu 60 sekund niekoniecznie można poznać rozwiązanie, jednak jeśli już to następuje jest to prawdopodobnie rozwiązanie jednoznaczne, z prawidłową sekwencją DNA podaną jako wynik. Tak szybkie odnajdywanie prawidłowej sekwencji jest zasługą konstrukcji chipu nieklasycznego, a dzięki niej możliwości weryfikowania wierzchołków na etapie tworzenia ścieżki. Dzięki temu nieprawidłowe rozgałęzienia są unikane, lub bardzo szybko algorytm wraca z nich na ścieżkę "prawidłową". Ponieważ kolejność ułożenia wierzchołków w zbiorze ma wpływ na kolejność układania listy wierzchołków do odwiedzenia w danym kroku, dla jakości prezentowanych rozwiązań bardzo ważne było jak najdokładniejsze wymieszanie elementów spektrum przed, a także po uwzględnieniu danego limitu błędów. Zostało to już podkreślone wcześniej, w podrozdziale opisującym szczegóły przeprowadzania testów.

Tablica 8.25: Chip typu Gapped *III* typu, zestawienie zbiorcze wyników prób z jednym, wieloma oraz prawidłowym rozwiązaniem

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	95(94)[95]	86(84)[86]	86(85)[85]	83(81)[83]	81(80)[81]
300 (10%)	94(91)[92]	86(80)[85]	81(76)[80]	88(85)[87]	70(67)[70]
300 (15%)	92(89)[92]	86(80)[85]	93(83)[92]	80(76)[76]	69(65)[67]
300 (20%)	88(84)[87]	91(88)[90]	87(82)[86]	83(77)[82]	75(66)[74]
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	92(91)[92]	77(77)[77]	74(74)[74]	66(65)[64]	68(68)[68]
500 (10%)	89(85)[88]	76(74)[76]	66(60)[64]	70(68)[68]	67(65)[63]
500 (15%)	90(88)[88]	84(82)[82]	69(64)[68]	72(69)[70]	64(63)[61]
500 (20%)	86(83)[86]	78(73)[76]	72(71)[69]	69(63)[68]	65(61)[64]
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	85(83)[85]	70(70)[70]	58(58)[58]	59(56)[58]	58(58)[57]
700 (10%)	67(63)[66]	69(66)[68]	65(63)[64]	63(58)[61]	60(58)[60]
700 (15%)	74(63)[72]	65(61)[65]	63(60)[62]	60(57)[58]	58(54)[56]
700 (20%)	71(67)[70]	63(55)[59]	62(60)[62]	56(54)[56]	50(47)[46]

Tablica 8.26: Chip typu Gapped *III* typu, zestawienie zbiorcze średniej liczby rozwiązań dla różnych wartości limitu błędów

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	1.01	1.02	1.01	1.04	1.01
300 (10%)	1.03	1.12	1.23	1.37	1.48
300 (15%)	1.03	1.10	1.68	1.05	1.10
300 (20%)	1.10	1.04	1.09	1.10	1.17
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	1.02	1.00	1.00	1.25	1.00
500 (10%)	1.04	1.05	1.22	1.20	1.85
500 (15%)	1.02	1.04	1.21	1.05	1.01
500 (20%)	1.03	1.06	1.01	1.24	1.07
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	1.04	1.05	1.00	1.11	1.00
700 (10%)	1.05	1.05	1.07	1.11	1.03
700 (15%)	1.20	1.12	1.15	1.08	1.07
700 (20%)	1.05	1.52	1.03	1.03	1.08

8.6.1.3 Alternating Chip

Kolejnym badanym chipem jest chip typu Alternating. Dwie pierwsze tabele, tj. 8.27 i 8.28 są analogiczne jak poprzednio, przedstawiają liczby prób z rozwiązaniami oraz średnią liczbę rozwiązań niejednoznacznych dla zaproponowanego w pracy algorytmu typu Alternating.

Z tabeli 8.27 i 8.28 widać, że algorytm dla spektrum pochodzącego z najmniejszego badanego w pracy rodzaju chipu typu Alternating ma duże trudności ze znalezieniem rozwiązań w zadanym czasie. Co prawda kiedy rozwiązanie jest w ciągu minuty podawane, jest ono najczęściej jednoznaczne oraz co ważniejsze, dla krótszych sekwencji niemal zawsze jest to prawidłowe badane DNA. Widać jednak, że mechanizm weryfikacji jest tutaj na tyle restrykcyjny, że w zadanym czasie algorytm ma jednak trudności z podaniem rozwiązania dla dłuższych fragmentów DNA. Trudno precyzyjnie wyrokować o ilości średniej liczby rozwiązań niejednoznacznych w porównaniu z dwoma poprzednimi chipami (8.20 i 8.24), ponieważ w omawianym przypadku bardzo niewiele prób kończyło się takim niejednoznacznym wynikiem - stąd podane wartości średnie należy traktować ostrożnie. Widać jednak wciąż ogromną przewagę nad podejściem klasycznym, oraz pewne podobieństwo do wyników algorytmu dla chipu typu Gapped.

Tablica 8.27: Chip typu Alternating *I* typu, zestawienie zbiorcze wyników prób z jednym, wieloma oraz prawidłowym rozwiązaniem

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	91(91)[91]	90(88)[89]	80(77)[80]	72(66)[70]	52(49)[52]
300 (10%)	75(66)[75]	65(50)[62]	64(54)[60]	55(47)[52]	58(53)[57]
300 (15%)	81(76)[81]	72(63)[69]	60(50)[59]	46(32)[40]	42(30)[40]
300 (20%)	77(67)[74]	59(41)[46]	57(39)[52]	56(38)[45]	40(26)[33]
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	72(69)[72]	61(59)[60]	26(24)[26]	14(12)[14]	6(6)[6]
500 (10%)	49(43)[49]	31(25)[30]	22(17)[20]	12(10)[12]	6(6)[6]
500 (15%)	48(39)[46]	26(19)[21]	16(10)[14]	14(13)[14]	8(5)[4]
500 (20%)	55(49)[54]	45(26)[34]	30(15)[22]	18(8)[13]	10(6)[8]
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	48(45)[48]	17(16)[17]	10(7)[10]	2(2)[2]	2(1)[2]
700 (10%)	24(23)[23]	13(11)[12]	2(1)[2]	4(1)[3]	1(0)[1]
700 (15%)	23(18)[19]	20(14)[15]	8(8)[5]	3(1)[2]	0(0)[0]
700 (20%)	23(21)[22]	17(8)[9]	6(1)[3]	2(1)[0]	3(0)[2]

Tablica 8.28: Chip typu Alternating *I* typu, zestawienie zbiorcze średniej liczby rozwiązań dla różnych wartości limitu błędów

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	1.00	1.01	1.04	1.08	1.13
300 (10%)	1.44	1.44	1.53	1.23	1.08
300 (15%)	1.09	1.33	1.31	1.67	1.57
300 (20%)	1.72	1.45	2.00	2.62	2.37
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	1.01	1.44	1.33	1.04	20.0
500 (10%)	1.28	1.19	1.50	1.25	1.00
500 (15%)	1.52	1.65	1.93	1.07	2.25
500 (20%)	1.21	4.04	7.90	6.94	5.40
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	1.02	1.05	1.16	1.00	1.50
700 (10%)	1.04	2.38	1.50	1.75	3.00
700 (15%)	1.65	1.85	1.00	2.00	0.00
700 (20%)	1.08	2.52	8.00	1.50	22.60

Kolejna tabela - 8.29 przedstawia analogiczne wyniki testów dla chipu Alternating *III* typu. Tabela 8.30 natomiast prezentuje, podobnie jak poprzednio, średnią liczbę rozwiązań.

Tablica 8.29: Chip typu Alternating *III* typu, zestawienie zbiorcze wyników prób z jednym, wieloma oraz prawidłowym rozwiązaniem

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	96(96)[96]	97(97)[97]	90(90)[89]	85(84)[85]	77(76)[77]
300 (10%)	84(79)[84]	92(86)[90]	85(81)[85]	87(84)[83]	84(83)[83]
300 (15%)	92(89)[92]	86(85)[86]	83(79)[83]	84(81)[82]	74(71)[73]
300 (20%)	89(86)[89]	86(81)[86]	84(80)[84]	81(79)[78]	70(69)[69]
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	85(84)[85]	85(84)[84]	79(78)[78]	68(65)[67]	63(59)[63]
500 (10%)	84(80)[84]	79(74)[79]	74(73)[72]	64(61)[63]	70(68)[70]
500 (15%)	83(83)[82]	77(75)[77]	73(71)[72]	64(59)[63]	53(52)[53]
500 (20%)	84(81)[84]	74(68)[73]	67(65)[67]	60(57)[60]	47(45)[47]
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	84(84)[83]	64(64)[63]	58(58)[58]	49(48)[49]	47(46)[47]
700 (10%)	74(74)[74]	57(55)[57]	51(48)[51]	42(41)[42]	40(37)[40]
700 (15%)	69(65)[66]	63(57)[62]	61(58)[58]	56(53)[54]	43(42)[43]
700 (20%)	70(66)[69]	68(65)[65]	53(49)[52]	53(52)[53]	52(49)[51]

Wyniki przedstawione w tabelach 8.29 oraz 8.30 są dość podobne do tych, które zaprezentowano dla algorytmu chipu Gapped *III* typu (8.25 oraz 8.26). Średnie liczby rozwiązań są do siebie bardzo zbliżone, jeśli natomiast mówimy o liczbie prób w zadanym czasie zakończonych sukcesem widać, że algorytm dla chipu typu Alternating znajduje ich trochę mniej niż poprzednik. Ich jakość jest jednak zbliżona i utrzymująca się na bardzo wysokim poziomie. Różnica pomiędzy próbami z rozwiązaniem, a rozwiązaniem jednoznacznym jest minimalna, a prawdopodobieństwo, że nawet w ciągu minuty algorytm poda prawidłową sekwencję jest bardzo wysokie.

8.6.1.4 Binary Chip

Podrozdział ten dotyczy ostatniego, trzeciego nieklasycznego chipu - typu Binary. Podobnie jak poprzednio przedstawione zostaną wyniki dla jego *I* i *III* typu. Tabele 8.31, 8.32, 8.33 oraz 8.34 prezentują wyniki w analogicznej kolejności jak dla poprzednich trzech algorytmów dla chipów: klasycznego, Gapped oraz Alternating.

Tablica 8.30: Chip typu Alternating *III* typu, zestawienie zbiorcze średniej liczby rozwiązań dla różnych wartości limitu błędów

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	1.00	1.00	1.00	1.10	1.02
300 (10%)	1.07	1.06	1.04	1.05	1.02
300 (15%)	1.05	1.01	1.16	1.23	1.04
300 (20%)	1.03	1.10	1.17	1.03	1.04
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	1.01	1.01	1.01	1.08	1.20
500 (10%)	1.08	1.25	1.01	1.06	1.07
500 (15%)	1.00	1.03	1.02	1.15	1.01
500 (20%)	1.03	1.10	1.02	1.05	1.04
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	1.00	1.00	1.00	1.04	1.02
700 (10%)	1.00	1.07	1.11	1.02	1.12
700 (15%)	1.11	1.19	1.14	1.05	1.04
700 (20%)	1.05	1.04	1.39	1.01	1.40

Porównując pierwsze dwie tabele odnoszące się do chipu Binary *I* typu (tabele 8.31, 8.32) z odpowiednikami dla chipów Gapped (tabele 8.23, 8.24) i Alternating(8.27, 8.28), widać ogromną różnicę na korzyść tego pierwszego. Algorytm dla chipu typu Binary w ciągu minuty jest w stanie z o wiele większym prawdopodobieństwem podać rozwiązanie, bardzo często jednoznaczne oraz niemal zawsze w zbiorze rozwiązań znajdzie się prawidłowa sekwencja. Szczególnie uderzające jest porównanie z największymi sekwencjami, zwłaszcza biorąc pod uwagę, że opisywane tabele dotyczą chipu najmniejszego (*I* typu). Algorytm dla chipu typu Gapped podaje wtedy więcej rozwiązań niejednoznacznych, z różnym prawdopodobieństwem znalezienia w nich prawidłowej sekwencji w zależności od parametrów testów. Dla najdłuższego testowanego DNA oraz największej liczby błędów w spektrum, w 60 sekund algorytm dla chipu typu Binary w około 20% prób podał rozwiązanie, najczęściej było to pojedyncze rozwiązanie jednoznaczne. Widać także, że różnica między liczbą prób z jakimkolwiek rozwiązaniem, a liczbą prób w których w rozwiązaniu była badana sekwencja (a najczęściej tylko ona), jest bardzo niewielka.

Jak wyjaśniono wcześniej, mechanizm weryfikacji wierzchołków dla chipu typu Binary zawarty jest niejako w samej strukturze wszystkich sond na powierzchni chipu, a nie jak w pozostałych przypadkach, tylko w jednej z jego połówek. Mechanizm ten jest tutaj o wiele precyzyjniejszy niż w chipach typu Alternating i Gapped, co łatwo zauważyć

porównując wyniki. Co ciekawe dla przeprowadzonych testów algorytm typu Binary jest o wiele mniej podatny na występowanie obu rodzajów błędów hybrydyzacji jednocześnie. W przypadku krótszych sekwencji, liczba prób z rozwiązaniem nie wzrasta wraz z limitem błędów, jak w poprzednich przypadkach dla algorytmów: klasycznego oraz typu Gapped (poza testami z sekwencją najdłuższą, mającą 700 par zasad). Widać, że im większy przyjęty limit, tym większe ryzyko wystąpienia rozwiązań niejednoznacznych, jednak co ciekawe, nie zmienia się stosunek znalezionych prawidłowych sekwencji do liczby wszystkich prób z rozwiązaniem. Wciąż pozostaje on zbliżony do liczby prób, które w ciągu minuty były w stanie podać jakiekolwiek rozwiązanie.

W tabelach 8.33 oraz 8.34 przedstawiających wyniki dla największego testowanego chipu Binary widać wyraźnie jego przewagę, gdyby porównać z innymi nieklasycznymi algorytmami. Generalnie istnieje bardzo wysokie prawdopodobieństwo otrzymania poszukiwanej sekwencji już w ciągu minuty obliczeń, niemal zawsze jednoznacznie. Przestawiono tutaj także tabele z wartościami średnimi liczby rozwiązań niejednoznacznych dla całego zbioru prób, należy jednak podchodzić do nich bardzo ostrożnie. Z różnic w liczbie prób z rozwiązaniem i rozwiązaniami jednoznacznymi widać, że bardzo niewiele kończyło się podaniem rozwiązań niejednoznacznych. Dlatego wyniki te mogłyby się różnić od przedstawionych, gdyby algorytm miał wystarczająco wiele czasu na przeszukanie całej przestrzeni rozwiązań.

Tablica 8.31: Chip typu Binary *I* typu, zestawienie zbiorcze wyników prób z jednym, wieloma oraz prawidłowym rozwiązaniem

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	96(95)[96]	92(92)[92]	87(86)[87]	82(82)[82]	80(80)[79]
300 (10%)	95(85)[93]	81(71)[79]	79(71)[78]	86(77)[83]	77(75)[77]
300 (15%)	94(85)[93]	85(75)[81]	85(74)[81]	79(70)[78]	75(67)[75]
300 (20%)	89(73)[88]	88(77)[88]	76(66)[76]	76(68)[75]	67(61)[64]
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	90(89)[90]	76(75)[76]	72(71)[72]	61(60)[60]	45(43)[43]
500 (10%)	75(67)[71]	60(54)[58]	53(50)[52]	46(38)[46]	46(43)[45]
500 (15%)	76(74)[74]	68(66)[67]	55(48)[55]	59(51)[53]	51(45)[50]
500 (20%)	79(71)[79]	66(58)[60]	57(54)[53]	49(45)[46]	39(30)[39]
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	76(76)[76]	58(57)[58]	43(42)[43]	30(30)[30]	20(19)[20]
700 (10%)	66(61)[64]	37(32)[35]	25(25)[24]	25(22)[22]	21(16)[21]
700 (15%)	58(53)[58]	35(30)[35]	37(29)[37]	29(25)[25]	21(16)[19]
700 (20%)	58(51)[58]	39(39)[36]	27(22)[24]	30(28)[30]	25(25)[24]

Tablica 8.32: Chip typu Binary *I* typu, zestawienie zbiorcze średniej liczby rozwiązań dla różnych wartości limitu błędów

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	1.01	1.00	1.02	1.00	1.00
300 (10%)	1.15	1.17	1.20	1.16	1.05
300 (15%)	1.12	1.14	1.21	1.13	1.13
300 (20%)	1.26	1.18	1.14	1.11	1.13
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	1.01	1.01	1.01	1.01	1.04
500 (10%)	1.12	1.10	1.05	1.17	1.06
500 (15%)	1.03	1.02	1.14	1.16	1.15
500 (20%)	1.11	1.21	1.05	1.10	1.46
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	1.00	1.01	1.09	1.00	1.05
700 (10%)	1.15	1.21	1.00	1.20	1.33
700 (15%)	1.12	1.14	1.24	1.93	1.23
700 (20%)	1.22	1.00	1.29	1.10	1.00

Tablica 8.33: Chip typu Binary *III* typu, zestawienie zbiorcze wyników prób z jednym, wieloma oraz prawidłowym rozwiązaniem

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	97(97)[97]	97(97)[97]	91(91)[90]	90(90)[90]	90(90)[90]
300 (10%)	99(98)[99]	92(89)[91]	93(91)[91]	88(86)[88]	89(89)[88]
300 (15%)	99(99)[99]	93(93)[92]	91(90)[91]	89(87)[89]	81(78)[79]
300 (20%)	100(99)[100]	97(94)[97]	92(89)[92]	90(88)[90]	90(89)[89]
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	95(95)[95]	90(89)[90]	82(82)[82]	82(81)[82]	72(70)[72]
500 (10%)	96(95)[96]	79(77)[79]	86(84)[85]	83(81)[82]	81(80)[81]
500 (15%)	97(97)[96]	85(83)[83]	89(87)[89]	79(78)[77]	79(78)[78]
500 (20%)	92(90)[91]	87(84)[85]	79(78)[77]	86(84)[84]	83(81)[79]
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	92(92)[92]	84(83)[84]	77(77)[77]	72(71)[72]	68(67)[68]
700 (10%)	86(86)[84]	85(83)[84]	87(85)[85]	80(79)[77]	65(64)[65]
700 (15%)	84(83)[83]	81(81)[78]	75(74)[74]	72(72)[71]	73(70)[70]
700 (20%)	87(86)[86]	81(78)[79]	77(75)[75]	72(68)[71]	74(71)[71]

Tablica 8.34: Chip typu Binary *III* typu, zestawienie zbiorcze średniej liczby rozwiązań dla różnych wartości limitu błędów

	Procent błędów hybrydyzacji				
Długość DNA (limit)	1%	2%	3%	4%	5%
DNA 300bp, błędy negatywne oraz limity					
300 (neg)	1.00	1.00	1.00	1.00	1.00
300 (10%)	1.04	1.03	1.03	1.02	1.00
300 (15%)	1.00	1.00	1.01	1.04	1.08
300 (20%)	1.01	1.06	1.06	1.02	1.03
DNA 500bp, błędy negatywne oraz limity					
500 (neg)	1.00	1.01	1.00	1.01	1.02
500 (10%)	1.01	1.02	1.08	1.02	1.01
500 (15%)	1.00	1.08	1.05	1.01	1.01
500 (20%)	1.03	1.03	1.01	1.03	1.02
DNA 700bp, błędy negatywne oraz limity					
700 (neg)	1.00	1.01	1.00	1.01	1.01
700 (10%)	1.00	1.04	1.05	1.01	1.04
700 (15%)	1.01	1.00	1.02	1.00	1.05
700 (20%)	1.01	1.03	1.06	1.11	1.09

Porównując wyniki z algorytmami typu Gapped i Alternating (nie wspominając już o zdecydowanie odstającym od nich podejściu klasycznym) widać wyraźnie przewagę chipu typu Binary. Średnia liczba rozwiązań niejednoznacznych jest zauważalnie mniejsza, natomiast co do ogólnej liczby prób zakończonych znalezieniem sekwencji DNA widać, że już dla najmniejszego testowanego chipu *I* typu algorytm jest w stanie częściej podać rozwiązanie niż jego odpowiedniki typu Gapped i Alternating.

8.6.2 Przypadek dokładnej wiedzy o liczbie błędów hybrydyzacji w spektrum

W całym poprzednim podrozdziale dotyczącym występowania w spektrum obu rodzajów błędów na raz testowane były trzy różne limity, które informowały algorytmy ile razy wolno im użyć wierzchołków nakładających się z poprzednikiem na długości mniejszej niż maksymalna dopuszczalna w każdym chipie. Limit ten, jak już napisano, ma bezpośredni wpływ na minimalną liczbę elementów spektrum, które algorytm musi użyć do rekonstrukcji DNA o zadanej długości, aby taką rekonstrukcję zaakceptować jako rozwiązanie dopuszczalne. Wyniki zaprezentowane w poprzednich tabelach wykazały znaczną przewagę podejścia nieklasycznego, które nawet przy wyjątkowo rozszerzonym

limicie błędów jest w stanie podać poprawne rozwiązania w bardzo krótkim czasie, a liczba rozwiązań niejednoznacznych jest znacznie mniejsza niż w przypadku podejścia klasycznego.

W ramach uzupełnienia podrozdziału dotyczącego błędów hybrydyzacji występujących równocześnie, poniżej zaprezentowane zostały tabele wyników dla chipów III typu, w których dana była wiedza o dokładnej liczbie błędów negatywnych. Algorytm wiedział więc z ilu precyzyjnie wierzchołków musi składać się rekonstrukcja DNA o danej długości. Błędy pozytywne występowały tak jak poprzednio. Tabela 8.35 prezentuje liczbę prób, w których algorytmy odnalazły rozwiązanie.

W tabeli 8.36 podano wartości średniej liczby rozwiązań dla tych samych testów:

Należy zaznaczyć, że do wyników tabeli 8.36 należy podchodzić bardzo ostrożnie. Jak widać z poprzedniej tabeli 8.35 niemal każda próba, która kończyła się podaniem rozwiązania w 60 sekund nie generowała w tak krótkim czasie rozwiązań niejednoznacznych. Oznacza to, że podane średnie dla wszystkich prób większe od 1.00 są rezultatem zazwyczaj jednej próby, która podawała pewną liczbę (niewielką, do maks 2 - 3) rozwiązań niejednoznacznych. Trudno więc na ich podstawie wyrokować o skuteczności. Taki przypadek (ze znajomością liczby błędów) lepiej przedstawia tabela 8.35, gdyż porównując ją z wynikami algorytmów dla chipów prezentowanych wcześniej - dla przybliżonego limitu, zawsze większego niż realna liczba błędów występujących w spektrum - widać dość wyraźnie, że taka hipotetyczna wiedza miałaby wpływ na jakość działania zaproponowanych algorytmów.

8.6.3 Wpływ czasu obliczeń na efektywność algorytmów

Wiele razy w poprzednich podrozdziałach poruszano kwestię czasu obliczeń. 60 sekund jest bardzo niewielkim limitem, jednak jak się okazuje, zaproponowane algorytmy nawet przy tak ekstremalnie ograniczonym czasie (biorąc pod uwagę rozmiary przestrzeni rozwiązań) były w stanie efektywnie radzić sobie z zadaniem sekwencjonowania nawet dłuższych sekwencji w obecności błędów hybrydyzacji.

W niniejszym podrozdziale zaprezentowano wyniki testów dla algorytmów chipów I typu, którym wydłużono dopuszczalny czas obliczeń do jednej godziny. Celem było sprawdzenie, jak zmienia się skuteczność algorytmów w kontekście zwiększania się liczby rozwiązań czy średniej liczby rozwiązań niejednoznacznych. Sprawdzana także była obecność rozwiązania prawidłowego (tj. badanej w danym teście sekwencji) w zbiorze rozwiązań wygenerowanych przez zaproponowane algorytmy.

Tabela 8.37 przedstawia wyniki testów w przypadku występowania obu rodzajów błędów hybrydyzacji.

Tablica 8.35: Liczba prób z rekonstrukcją DNA dla chipów przy wiedzy o liczbie błędów negatywnych (błędy negatywne i pozytywne)

	Procent błędów hybrydyzacji				
DNA	1%	2%	3%	4%	5%
Chip klasyczny I i III typu					
300	78(34)	55(39)	39(27)	22(19)	22(20)
500	24(15)	13(8)	8(6)	4(2)	5(1)
700	10(7)	3(2)	4(3)	2(0)	4(0)
300	94(86)	79(77)	75(73)	73(71)	73(72)
500	74(70)	59(57)	43(42)	38(38)	42(41)
700	37(36)	44(43)	27(26)	22(22)	28(27)
Gapped Chip I i III typu					
300	91(87)	78(72)	66(59)	62(57)	66(54)
500	61(53)	52(44)	44(31)	33(26)	35(19)
700	45(37)	21(18)	25(15)	9(5)	7(6)
300	98(98)	84(84)	81(81)	86(85)	78(74)
500	82(82)	75(74)	71(71)	65(64)	62(59)
700	78(78)	72(69)	65(64)	64(64)	62(62)
Alternating Chip I i III typu					
300	90(90)	81(80)	85(81)	72(68)	52(46)
500	79(78)	54(47)	33(30)	23(22)	4(3)
700	41(40)	20(19)	6(6)	3(3)	0(0)
300	96(95)	88(87)	91(90)	83(82)	78(76)
500	89(88)	77(75)	81(80)	66(63)	68(67)
700	80(78)	72(72)	58(57)	48(46)	52(51)
Binary Chip I i III typu					
300	98(98)	92(91)	88(88)	85(85)	77(75)
500	87(87)	80(80)	69(68)	58(58)	61(60)
700	75(73)	64(63)	55(54)	48(48)	48(47)
300	100(100)	96(96)	88(87)	88(87)	88(88)
500	96(96)	89(89)	82(82)	81(80)	73(71)
700	88(88)	79(79)	81(80)	75(75)	68(68)

Tablica 8.36: Średnia liczba rozwiązań dla chipów przy wiedzy o liczbie błędów negatywnych (błędy negatywne i pozytywne)

	Procent błędów hybrydyzacji				
DNA	1%	2%	3%	4%	5%
Chip klasyczny *I* i *III* typu					
300	7.29	1.61	1.53	1.63	1.40
500	1.66	1.38	2.50	2.25	7.80
700	1.30	1.33	1.25	5.00	8.75
300	1.10	1.03	1.02	1.05	1.01
500	1.08	1.06	1.02	1.00	1.02
700	1.02	1.06	1.03	1.00	1.03
Gapped Chip *I* i *III* typu					
300	1.05	1.10	2.27	1.09	1.34
500	1.18	1.18	1.54	1.30	1.32
700	1.20	1.22	1.36	1.33	1.14
300	1.00	1.00	1.00	1.01	1.05
500	1.00	1.01	1.00	1.01	1.08
700	1.00	1.04	1.01	1.00	1.00
Alternating Chip *I* i *III* typu					
300	1.00	1.01	1.04	1.08	1.13
500	1.01	1.44	1.33	1.04	1.00
700	1.02	1.05	1.16	1.00	1.00
300	1.01	1.03	1.01	1.01	1.11
500	1.01	1.05	1.11	1.07	1.20
700	1.02	1.00	1.03	1.04	1.03
Binary Chip *I* i *III* typu					
300	1.00	1.01	1.00	1.00	1.02
500	1.00	1.00	1.01	1.00	1.01
700	1.03	1.03	1.05	1.01	1.06
300	1.00	1.00	1.01	1.01	1.00
500	1.00	1.00	1.00	1.02	1.05
700	1.00	1.00	1.02	1.00	1.00

Tablica 8.37: Zmiany liczby rozwiązań przy zwiększonym czasie przeszukiwania (wszystkie typy błędów)

Typ	>	=	DNA?	średnio.	Typ	>	=	DNA?	średnio.
Gapped Chip					Alternating Chip				
0 rozw.	40%	60%	30%	2.50	0 rozw.	30%	70%	30%	1.00
1 rozw.	20%	80%	80%	2.00	1 rozw.	0%	100%	100%	1.00
wiele	50%	50%	70%	3.50	wiele	20%	80%	20%	3.00
Binary Chip					Chip klasyczny				
0 rozw.	40%	60%	40%	1.00	0 rozw.	100%	0%	10%	98.40
1 rozw.	10%	90%	100%	2.00	1 rozw.	100%	0%	10%	101.40
wiele	20%	80%	100%	2.50	wiele	100%	0%	20%	133.50

Parametry testów, których wyniki przedstawia wspomniana tabela są następujące:

1) długość DNA równa 300 par zasad;

2) od 1% do 3% błędów w spektrum;

3) chipy I typu;

4) oba rodzaje błędów hybrydyzacji;

5) przyjęty przybliżony limit błędów równy 15% liczby elementów spektrum idealnego;

6) czas obliczeń równy 3600 sekund.

Dla każdego chipu wybrano losowo 30 instancji, które wcześniej brały udział w badaniach opisanych w poprzednich podrozdziałach, dotyczących obu rodzajów błędów. Instancje te dzieliły się na trzy równe grupy po 10, w zależności od wyników opisanych poprzednio. Pierwsza grupa dawała wcześniej w czasie 60 sekund zero rozwiązań, druga - tylko jedno, trzecia natomiast grupa to taka, dla której algorytmy w poprzednich testach generowały rozwiązania niejednoznaczne. Następnie algorytmy z nowym limitem czasu były testowane dla tych wybranych instancji, zgodnie z parametrami testów danymi powyżej.

Prezentowana tabela 8.37 dzieli się na 4 ćwiartki, każda dla odpowiedniego chipu. W każdej ćwiartce znajdują się 3 wiersze oraz 5 kolumn. Wiersze te dotyczą trzech opisanych już grup instancji pobranych z poprzednich testów. Pierwsza kolumna opisuje grupę dziesięciu fragmentów DNA. Druga i trzecia kolumna to procent odpowiednio: liczby prób, które dały więcej rozwiązań przy powiększonym czasie obliczeń oraz procent prób, dla których w ciągu godziny liczba rozwiązań podanych przez algorytm nie

zmieniła się. Czwarta kolumna to procent prób (zawsze z dziesięciu), w których w zbiorze rozwiązań znalazło się prawidłowe DNA. Ostatnia, piąta kolumna dla danego chipu pokazuje średnią liczbę rozwiązań tych prób, które generowały rozwiązania niejednoznaczne.

Należy zauważyć, że w przypadku instancji, które wcześniej generowały tylko jedno rozwiązanie, trzy algorytmy dla chipów nieklasycznych po godzinie kończyły pracę z tą samą liczbą rozwiązań. Dla chipu typu Binary tylko jedna instancja zakończyła się podaniem rozwiązań niejednoznacznych, dwie dla algorytmu chipu Gapped, zero w przypadku chipu Alternating. Jeśli chodzi o algorytm klasyczny, zawsze po godzinie kończył on pracę podając rozwiązania niejednoznaczne. Lepiej w tym zestawieniu wypadły algorytmy dla chipów Alternating i Binary. Zwłaszcza ten ostatni ma bardzo wysoki procent prób, w których w zbiorze rozwiązań znajdowało się badane DNA. Dla instancji testowych z jednym rozwiązaniem dobrze prezentuje się algorytm chipu Alternating. Jak widać z tabeli, nawet w ciągu godziny nie zaczął on generować rozwiązań niejednoznacznych. Należy tutaj podkreślić, że omawiany jest przypadek, w którym przybliżony limit błędów wynosi 15%, co oznacza, że rozwiązania akceptowalne muszą wykorzystać tylko 85% liczby elementów spektrum idealnego. Nawet wtedy mechanizm weryfikacji jest na tyle restrykcyjny, że nie dopuszczał do generowania takich rozwiązań.

Druga zaprezentowana tabela - 8.38, różni się tylko jednym parametrem testów od poprzedniej. W tym wypadku brano pod uwagę tylko instancje z błędami negatywnymi.

Tablica 8.38: Zmiany liczby rozwiązań przy zwiększonym czasie przeszukiwania (tylko błędy negatywne)

Typ	>	=	DNA?	średnio	Typ	>	=	DNA?	średnio
Gapped Chip					Alternating Chip				
0 rozw.	40%	60%	30%	1.00	0 rozw.	40%	60%	40%	1.25
1 rozw.	0%	100%	100%	1.00	1 rozw.	0%	100%	100%	1.00
wiele	20%	80%	80%	2.10	wiele	10%	90%	100%	2.80
Binary Chip					Chip klasyczny				
0 rozw.	60%	40%	60%	1.00	0 rozw.	40%	60%	40%	1.60
1 rozw.	0%	100%	100%	1.00	1 rozw.	70%	30%	10%	1.40
wiele	20%	80%	100%	2.50	wiele	100%	0%	60%	5.40

Jak widać z prezentowanych w niej wyników, wzrosła liczba przypadków, dla których algorytmy dla chipów nieklasycznych Alternating i Binary generowały rozwiązanie, gdy w 60 sekundach nie były w stanie tego zrobić. Zauważalnie spadła też średnia liczba rozwiązań niejednoznacznych w porównaniu z przypadkiem obu rodzajów błędów w

spektrum. W tym zestawieniu wciąż najlepiej prezentują się wyniki dla algorytmów Alternating i Binary.

8.6.4 Porównanie efektywności algorytmów z innymi podejściami do metody SBH

W tym podrozdziale zostaną przedstawione wyniki testów, które przeprowadzono w celu wykazania skuteczności zaprojektowanych algorytmów, w porównaniu z innymi metodami opublikowanymi w literaturze naukowej, na zbiorze tych samych sekwencji. Poniżej przytoczono część tabeli wyników z artykułu [9], w którym testowano między innymi hybrydowy algorytm genetyczny do problemu klasycznego SBH z błędami. Użyto w tym celu zbioru 40 sekwencji, które autorzy udostępnili do dalszych badań i testów. Pierwsza część tabeli 8.39 zawiera wyniki algorytmu genetycznego, jej druga część to wyniki z publikacji [8], w której testowano algorytm poszukiwania tabu (ang. Tabu Search) oparty na sekwencjonowaniu z użyciem izotermicznych bibliotek oligonukleotydów. Podejście to zostało pokrótce przedstawione w rozdziale poświęconym różnym wersjom metody SBH.

Tablica 8.39: Wyniki algorytmów przybliżonych

	Długość DNA					
	200		400		600	
	Procent błędów hybrydyzacji					
	5%	20%	5%	20%	5%	20%
PC Pentium 4, 2.0 GHz, 512MB RAM						
Poprawiony hybrydowy algorytm genetyczny [9]						
Wykorzystanie sond %	100	100	100	100	100	99.98
Podobieństwo %	100	99.88	100	100	100	99.98
Liczba rozwiązań optym.	40/40	39/40	40/40	40/40	40/40	37/40
Czas obliczeń	4.30	6.32	11.90	20.46	25.69	44.19
Algorytm Tabu Search [8]						
Wykorzystanie sond %	99.93	99.93	99.90	99.67	99.84	99.36
Podobieństwo %	99.87	98.44	98.96	95.70	95.82	88.50
Liczba rozwiązań optym.	39/40	38/40	37/40	28/40	32/40	19/40
Czas obliczeń	5.62	8.99	43.38	68.50	127.57	186.26

Jak widać z tabeli 8.39, algorytmy generują rozwiązania o pewnym stopniu podobieństwa do sekwencji badanej. Kryterium rozwiązania jest też procent użycia elementów ze spektrum. Dla tego samego zbioru 40 sekwencji, o długościach 200, 400 i 600 par zasad przeprowadzono testy z użyciem zaprojektowanych i opisanych w rozprawie algorytmów. Przyjęto podejście oparte na chipach III typu. Testy, których wyniki zaprezentowano

w tabeli 8.39 przeprowadzono dla oligonukleotydów o długości $l = 10$ w przypadku algorytmu genetycznego (odpowiednik właśnie chipu nieklasycznego III typu wg. notacji stosowanej w niniejszej rozprawie). Algorytm przeszukiwania tabu wykorzystywał podejście oparte na bibliotekach oligonukleotydów izotermicznych, o parametrach temperatury: $t = 26$ i $t+2 = 28$. Wyniki dla algorytmów dedykowanych chipom nieklasycznym przedstawia tabela 8.40.

Tablica 8.40: Wyniki algorytmów dla chipów nieklasycznych

	Długość DNA								
	200			400			600		
	Procent błędów hybrydyzacji								
	5%	10%	20%	5%	10%	20%	5%	10%	20%
Gapped Chip									
Prawidłowe:	36/40	34/40	12/40	35/40	31/40	5/40	36/40	28/40	4/40
1 rozwiąz.:	35/40	30/40	12/40	34/40	30/40	5/40	34/40	26/40	4/40
Wiele :	1/40	4/40	0/40	1/40	1/40	1/40	2/40	2/40	0/40
Czas [s]	60	300	300	60	300	300	60	300	300
Alternating Chip									
Prawidłowe:	36/40	24/40	7/40	32/40	23/40	2/40	32/40	14/40	0/40
1 rozwiąz.:	35/40	24/40	6/40	32/40	23/40	2/40	32/40	14/40	0/40
Wiele :	1/40	0/40	1/40	1/40	0/40	0/40	0/40	0/40	0/40
Czas [s]	60	300	300	60	300	300	60	300	300
Binary Chip									
Prawidłowe:	37/40	30/40	16/40	37/40	30/40	18/40	34/40	33/40	6/40
1 rozwiąz.:	37/40	30/40	16/40	37/40	30/40	18/40	34/40	32/40	6/40
Wiele :	0/40	0/40	0/40	0/40	0/40	0/40	0/40	1/40	0/40
Czas [s]	60	300	300	60	300	300	60	300	300

Dla testów z 20% błędów negatywnych i pozytywnych w spektrum, limit przyjmowany przez algorytm został rozszerzony do 50%. Niestety tak "zabrudzone" błędami spektra okazały się bardzo trudnymi przypadkami dla zaproponowanych algorytmów. Algorytmy nie były w stanie w ciągu 5 minut podać rozwiązania dla wielu sekwencji testowych. Tak wysoka liczba błędów powoduje, że przestrzeń rozwiązań staje się po prostu zbyt duża, a zaproponowane podejście przestaje być efektywne, nawet z użyciem chipów nieklasycznych.

Z porównania tabel 8.39 i 8.40 widać jednak, że podejście stosowane w zaprojektowanych algorytmach dla chipów nieklasycznych prezentuje się dość dobrze jeżeli liczba błędów (oddzielnie dla obu połówek chipu) nie przekracza 10%. Całkowita liczba instancji z rozwiązaniem jest mniejsza, niż w przypadku algorytmu genetycznego, lecz mowa tutaj o precyzyjnie zrekonstruowanej sekwencji. Nie ma tutaj mowy o stopniu

podobieństwa do badanego DNA, ponieważ dla 5% błędów w spektrum zdecydowana większość testów podawała w ciągu 300 sekund jednoznaczne rozwiązanie, które okazywało się badanym DNA. Liczba prób z rozwiązaniem niejednoznacznym okazała się być minimalna, i tylko ta wartość mogłaby tutaj odpowiadać za pojawianie się sekwencji "podobnych" do badanego DNA. W przypadku zwiększenia liczby błędów, zaprojektowane algorytmy wciąż są w stanie pracować z dobrą skutecznością. Jej miarą jest liczba rozwiązań jednoznacznych oraz obecność w zbiorze rozwiązań poszukiwanej sekwencji.

Dla zbioru wymienionych 40 sekwencji przeprowadzono także specjalne, oddzielne testy, w których w spektrum 5%, 10% i 20% błędów hybrydyzacji występuje oddzielnie, tj. na raz tylko jeden typ błędów był obecny. Wyniki zaprezentowano w tabeli 8.41. Wyraźnie widać, że błędy pozytywne nie mają wpływu na efektywność algorytmów. Należy dodać, że dla dwóch instancji, w których algorytm dla chipu Alternating nie był w stanie wyszukać rozwiązania, pojawiły się nieliczne błędy negatywne wynikające z powtórzeń. Sekwencje tutaj używane były przed autorów wcześniejszych wymienionych publikacji wybrane jako nie posiadające tego rodzaju błędów *w podejściu klasycznym*. Ponieważ budowa sond w chipach nieklasycznych omawianych w rozprawie znacznie się od tego podejścia różni, błędy wynikające z powtórzeń wciąż mogły wystąpić. W dwóch przypadkach (próbach z 40) dla algorytmu chipu typu Alternating taka sytuacja miała miejsce, stąd wynik 38/40 prób zakończonych sukcesem. Co do błędów negatywnych, widać wyraźnie, jak duży wpływ ma ich liczba na efektywność algorytmów. Dla ich liczby równej 20% w zasadzie tylko algorytm dla chipu typu Binary był w stanie w pewnej części prób wygenerować rozwiązanie. W przypadku algorytmów dla chipów typu Gapped i Alternating, odpowiednio dwie i jedna próba na zbiorze sekwencji zakończyły się znalezieniem badanego DNA. W porównaniu z poprzednią tabelą widać całkowity brak rozwiązań niejednoznacznych (w zadanym czasie). O ile w tabeli 8.40 występowały one niezmiernie rzadko, tak w przypadku, gdy występował tylko jeden rodzaj błędów nie pojawiły się one w przypadku pracy nad żadną sekwencją testową. Wynika to z możliwości precyzyjnego określenia liczby elementów ze spektrum potrzebnych do rekonstrukcji DNA.

8.6.5 Podsumowanie wyników testów obliczeniowych

Wszystkie przeprowadzone testy zdają się potwierdzać potencjał zawarty w nieklasycznej budowie rozpatrywanych w rozprawie chipów do sekwencjonowania przez hybrydyzację. Opracowane algorytmy radzą sobie dobrze, nawet pracując na instancjach (spektrach) obarczonych wysoką liczbą błędów hybrydyzacji. Oba rodzaje tych błędów występujących jednocześnie w spektrum mają oczywiście negatywny wpływ na jakość wyników, jednak wciąż są one wystarczająco dobre aby uzasadniać użycie takiego podejścia do

Tablica 8.41: Wyniki algorytmów dla chipów nieklasycznych, z błędami występującymi oddzielnie

	Długość DNA					
	600		600		600	
	Procent błędów hybrydyzacji					
	+5%	−5%	+10%	−10%	+20%	−20%
Gapped Chip						
Znaleziono prawidłowe DNA	40/40	36/40	40/40	29/40	40/40	2/40
Tylko 1 rozwiązanie	40/40	36/40	40/40	29/40	40/40	2/40
Wiele rozwiązań w zbiorze	0/40	0/40	0/40	0/40	0/40	0/40
Czas obliczeń [s]	60	120	60	300	60	300
Znaleziono prawidłowe DNA	38/40	33/40	38/40	17/40	38/40	1/40
Tylko 1 rozwiązanie	38/40	33/40	38/40	17/40	38/40	1/40
Wiele rozwiązań w zbiorze	0/40	0/40	0/40	0/40	0/40	0/40
Czas obliczeń [s]	60	120	60	300	60	300
Znaleziono prawidłowe DNA	40/40	37/40	40/40	33/40	40/40	16/40
Tylko 1 rozwiązanie	40/40	37/40	40/40	33/40	40/40	16/40
Wiele rozwiązań w zbiorze	0/40	0/40	0/40	0/40	0/40	0/40
Czas obliczeń [s]	60	120	60	300	60	300

problemu sekwencjonowania. Porównując zaproponowane algorytmy z innymi, opublikowanymi w literaturze naukowej widać, że radzą sobie jak najbardziej zadowalająco, bardzo często generując w krótkim czasie rozwiązanie jednoznaczne, będące badanym DNA. Liczba rozwiązań niejednoznacznych jest niewielka, natomiast liczba prób, w których w zadanym czasie opracowane algorytmy nie generowały prawidłowego DNA jest minimalna, w stosunku do liczby prób z rozwiązaniem.

Rozdział 9

Podsumowanie

Sekwencjonowanie DNA stanowi podstawowy problem, który musi być rozwiązywany w ramach wielu współcześnie prowadzonych badań nad organizmami żywymi. Poznanie sekwencji DNA danego organizmu stanowi zawsze początek dalszych prac w biologii molekularnej, genetyce oraz w wielu innych współcześnie przeprowadzanych badaniach zmierzających do poznania procesów zachodzących w żywych komórkach. Istnieje wiele metod poznania struktury DNA, od najstarszych, wciąż gdzieniegdzie stosowanych, wymagających czasu i skomplikowanych przygotowań, po najnowsze osiągnięcia w dziedzinie sekwencjonowania jak sekwenatory DNA precyzyjnie wykrywające dołączenie kolejnych nukleotydów w analizowanej sekwencji.

Metoda sekwencjonowania przez hybrydyzację stanowi odmianę sekwencjonowania, łączącą w sobie zarówno podejście biochemiczne, tj. eksperyment hybrydyzacyjny z udziałem chipu DNA, z podejściem obliczeniowym, związanym z konstrukcją algorytmu pracującego na spektrum DNA często obarczonym błędami. Metoda ta liczy sobie już ponad 20 lat, lecz pomimo obecności nowszych podejść, nadal jest badana i udoskonalana, czego dowodem mogą być wymienione w rozdziale 4 jej liczne odmiany. Stosowane są różne podejścia mające na celu udoskonalenie metody SBH. Część z nich koncentruje się na zmianach w konstrukcji chipu DNA, jak np. używanie nie wszystkich, lecz wyselekcjonowanych oligonukleotydów w sondach. Inne polegają np. dodatkowych eksperymentach mających na celu uzyskanie danych o przybliżonej lokalizacji w DNA podciągu, który hybrydyzował z daną sondą. Jeszcze inne z kolei podejścia dotyczą części obliczeniowej metody. Powstają wciąż nowe algorytmy przybliżone, mogące w bardzo krótkim czasie rekonstruować sekwencje DNA ze spektr obarczonych nawet bardzo dużym procentem błędów hybrydyzacji. Wadą takiego podejścia jest jednak to, że w przypadku dłuższych fragmentów badanego DNA często otrzymujemy rekonstrukcję przybliżoną, nie będącą identyczną z badanym DNA.

Badane w niniejszej rozprawie doktorskiej podejście łączy w sobie zarówno zmiany

w sposobie konstrukcji chipu DNA, jak i konieczność opracowania całkowicie nowych algorytmów, które operować muszą na spektrum znacznie różniącym się od tego z podejścia klasycznego. Zmiany w fazie biochemicznej SBH wymagają stosowania nowych związków chemicznych, jak np. opisane już wcześniej nukleotydy uniwersalne czy tak zwane nukleotydy zdegenerowane. Z drugiej strony, można obejść ten wymóg stosując wszystkie kombinacje oligonukleotydów w sondzie, które pasują do wzoru ją opisującego. Takie podejście stosować można tylko do pewnego stopnia, ponieważ ma ono wpływ na zmniejszanie siły sygnału hybrydyzacji, czyli "śladu" po połączeniu DNA z sondą, który odpowiada za możliwość wyselekcjonowania sond chipu po eksperymencie do ostatecznego spektrum DNA.

Znaczna część niniejszej rozprawy poświęcona została zaproponowanym algorytmom. Ich idea zbliżona jest do wyszukiwania dokładnego, jednak wykorzystuje ona także dodatkowe informacje, umożliwiające skuteczną weryfikację kolejnych kroków rekonstrukcji DNA. Algorytmy te są też do pewnego stopnia odporne na błędy hybrydyzacji. Ciekawym przykładem jest spektrum pochodzące z chipu typu Binary. Każda sonda chipu może być traktowana jako zbiór oligonukleotydów, do każdej sondy w eksperymencie hybrydyzacyjnym może hybrydyzować wiele różnych fragmentów badanego DNA. Zaproponowany algorytm radzi sobie z tym nad wyraz dobrze, co wykazały liczne przeprowadzone testy. Weryfikacja połączona tutaj z warunkiem zgodności ostatniego nukleotydu następników w ścieżkach grafów pochodzących z dwóch połówek spektrum sprawuje się wyjątkowo dobrze. Pozwala ona z dużym prawdopodobieństwem sekwencjonować dłuższe fragmenty DNA, nawet ze spektrum obarczonego też innymi rodzajami błędów.

W przypadku zaproponowanych dla omawianych chipów algorytmów możliwe są dalsze usprawnienia, w toku przyszłych prac. Jedną z możliwości jest wykorzystanie podejścia z cytowanej w rozdziale 4. literatury, polegającego na częściowym usuwaniu błędów pozytywnych w spektrum. Pomimo używania tutaj podejścia nieklasycznego do projektowania chipów, elementy spektrum są od siebie zależne w związku z wzajemnym nakładaniem. Dlatego wciąż istnieje możliwość wykorzystania tego faktu w procesie usuwania choćby części błędów pozytywnych. Zaproponowane tu podejście do konstrukcji algorytmu wykazało w testach swoją skuteczność, jednak wciąż nie ulega wątpliwości, że jest to rozwiązanie bazujące na częściowym przeszukiwaniu całej przestrzeni rozwiązań. Możliwe do realizacji są także inne podejścia i typy algorytmów, starające się jeszcze efektywniej wykorzystywać informacje dane w odpowiednich spektrach.

Podsumowując temat algorytmów dla omawianych chipów można wspomnieć o wcześniejszych próbach ich zaprojektowania, tym razem w wariancie tylko z błędami negatywnymi wynikającymi z powtórzeń. W pracy [45] opisano algorytmy dla chipów typu Gapped i Binary, pracujące na spektrach tylko z takimi błędami. Wyniki okazały się bardzo obiecujące, ponieważ algorytmy były w stanie precyzyjnie sekwencjonować DNA nawet o długościach przekraczających cztery tysiące par zasad. Co jeszcze ciekawsze,

wyniki pokazały skuteczność algorytmów dla spektrum z chipu typu Binary, posiadającego o jeszcze jeden stopień mniej sond niż testowany tu najmniejszy chip I typu, precyzyjniej chipu dwukrotnie mniejszego. Stosując przyjęty w niniejszej pracy podział, dla chipów I typu oba opisane w cytowanej pracy algorytmy miały niemal 100% skuteczność w jednoznacznej rekonstrukcji DNA o długości dochodzącej do 5000 par zasad. Należy jednak podkreślić, że algorytmy te pracowały dla hipotetycznego spektrum, w którym jedyne błędy negatywne wynikały z samej budowy DNA. Założono więc teoretyczną, perfekcyjną jakość procesu hybrydyzacji, który nie generuje w trakcie trwania żadnych nowych błędów. Badania z niniejszej rozprawy oraz te opisane w przytoczonym artykule dowodzą ogromnego potencjału omawianych tutaj chipów.

Kolejnym zagadnieniem, któremu poświęcono wiele miejsca w niniejszej rozprawie doktorskiej jest złożoność obliczeniowa problemów związanych z chipami nieklasycznymi oraz odpowiednimi typami błędów. Do tej pory w literaturze znana jest przynależność sekwencjonowania przez hybrydyzację w wersji klasycznej z błędami, do klasy problemów silnie NP-trudnych [10]. W rozdziale omawiającym różne podejścia do SBH wymieniono także publikacje związane z tzw. pozycyjnym SBH, którego złożoność obliczeniowa w kilku wariantach również została ustalona ([24], [2]). Do tej pory omawiane chipy były stosunkowo rzadko rozpatrywane w literaturze, zaś sama idea nukleotydu uniwersalnego (znak X w chipach) była rozpatrywana głównie z punktu widzenia algorytmów lub innych sposobów jego wykorzystania do konstrukcji sond, niż te przyjęte w chipach Gapped i Alternating. W niniejszej rozprawie sformułowano problemy decyzyjne oraz przeszukiwania dla wszystkich trzech chipów, w wersjach bez błędów hybrydyzacji, z błędami negatywnymi oraz błędami pozytywnymi. Praca zawiera także dowody złożoności obliczeniowej dla problemów z błędami pozytywnymi, dla wszystkich badanych chipów. Może stanowić to dobry punkt wyjścia dla badań nad złożonością obliczeniową tych wariantów problemów, których złożoność nie została jeszcze określona.

Wyniki przedstawione w niniejszej rozprawie dowodzą dużego potencjału omawianych tu chipów nieklasycznych, przede wszystkim związanego z nimi mechanizmu weryfikacji następników w toku pracy odpowiednich, dedykowanych im algorytmów. Mechanizm ten pozwala uzyskać rozwiązanie z dużym prawdopodobieństwem, że jest to rozwiązanie oczekiwane (tj. prawdziwa badana sekwencja DNA), nawet w krótkim czasie. Opracowane algorytmy wykazują też odporność na różne błędy hybrydyzacji w spektrum. Przy wystąpieniu wszystkich ich rodzajów jednocześnie wyniki wciąż są satysfakcjonujące, zwłaszcza w przypadku chipu typu Binary. Modyfikacje algorytmów lub opracowanie ich w wersji przybliżonej mogą doprowadzić do lepszych rezultatów, a na pewno stanowią bardzo ciekawy przedmiot dalszych badań.

Bibliografia

[1] W. Bains, G. C. Smith. A novel method for nucleic acid sequence determination. *Journal of Theoretical Biology*, 135:303–307, 1988.

[2] A. Ben-Dor, I. Pe'er, R. Shamir, R. Sharan. On the complexity of positional sequencing by hybridization. *Journal of Computational Biology*, 8:361–371, 2001.

[3] M. Berger, Y. Wu, A. Ogawa, D. McMinn, P. Schultz, F. Romesberga. Universal bases for hybridization, replication and chain termination. *Nucleic Acids Research*, 28:2911–2914, 2000.

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

[5] J. Blazewicz, P. Formanowicz. Multistage isothermic sequencing by hybridization. *Computational Biology and Chemistry*, 29:69–77, 2005.

[6] J. Blazewicz, P. Formanowicz, M. Kasprzak, W. T. Markiewicz. Sequencing by hybridization with isothermic oligonucleotide libraries. *Discrete Applied Mathematics*, 145:40–51, 2004.

[7] J. Blazewicz, P. Formanowicz, M. Kasprzak, W. T. Markiewicz, J. Węglarz. DNA sequencing with positive and negative errors. *Journal of Computational Biology*, 6:113–123, 1999.

[8] J. Blazewicz, P. Formanowicz, M. Kasprzak, W. T. Markiewicz, A. Świercz. Tabu search algorithm for DNA sequencing by hybridization with isothermic libraries. *Computational Biology and Chemistry*, 28:11–19, 2004.

[9] J. Blazewicz, F. Glover, M. Kasprzak, W. T. Markiewicz, C. Oguz, D. Rebholz-Schuhmann, A. Świercz. Dealing with repetitions in sequencing by hybridization. *Computational Biology and Chemistry*, 30:313–320, 2006.

[10] J. Blazewicz, M. Kasprzak. Complexity of DNA sequencing by hybridization. *Theoretical Computer Science*, 290:1459–1473, 2003.

[11] C. Blum, M. Vallcs, M. Blesa. An ant colony optimization algorithm for DNA sequencing by hybridization. *Computers & Operations Research*, 35:3620 – 3635, 2008.

[12] S. Cook. The complexity of theorem proving procedures. *Proceedings of 3rd ACM Symposium Theory of Computing*, strony 151–158, 1971.

[13] F. Crick. On protein synthesis. *Symposia of the Society for Experimental Biology*, XII:139–163, 1958.

[14] F. Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[15] M. Dorigo, T. Stützle. *Ant colony optimization*. MIT Press, 2004.

[16] R. Drmanac, L. Labat, I. Brukner, R. Crkvenjakov. Sequencing of megabase plus DNA by hybridization. *Genomics*, 4:114–128, 1989.

[17] T. A. Endo. Probabilistic nucleotide assembling method for sequencing by hybridization. *Bioinformatics*, 20:2181–2188, 2004.

[18] O. Fedrigo, G. Naylor. A gene-specific DNA sequencing chip for exploring molecular evolutionary change. *Nucleic Acids Research*, 32:1208–1213, 2004.

[19] P. Formanowicz. DNA sequencing by hybridization with additional information available. *Computational Methods in Science and Technology*, 11(1):21–29, 2005.

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

[21] P. Formanowicz. Isothermic sequencing by hybridization problems with information about repetitions. *Electrical Review*, 9:103–107, 2008.

[22] A. Frieze, B. Halldorsson. Optimal sequencing by hybridization in rounds. *Journal of Computational Biology*, 9:355–369, 2001.

[23] J. Gallego, D. Loakes. Solution structure and dynamics of DNA duplexes containing the universal base analogues 5-nitroindole and 5-nitroindole 3-carboxamide. *Nucleic Acids Research*, 35:2904–2912, 2007.

[24] S. Hannenhalli, P. A. Pevzner, H. Levis, S. Skiena. Positional sequencing by hybridization. *Computer Applications in Biosciences*, 12:19–24, 1996.

[25] D. Horspool. Extended central dogma with enzymes. *http://en.wikipedia.org/wiki/File Extended_Central_Dogma_with_Enzymes.jpg*. Wikimedia Commons, 2008.

[26] D. S. Johnson. The np-completeness column: an ongoing guide. *Journal of Algorithms*, 6:291–305, 1985.

[27] K. R. Khrapko, P. Lysov, A. A. Khorlyn, V. V. Shick, V. L. Florentiev, A. D. Mirzabekov. An oligonucleotide hybridization approach to DNA sequencing. *FEBS Letters*, 256:118–122, 1989.

[28] K. Kwarciak, M. Radom, P. Formanowicz. Sekwencjonowanie DNA z błędami negatywnymi oraz informacją o powtórzeniach. *Zeszyty Naukowe Politechniki Śląskiej*, z. 151:215–222, 2008.

[29] P. M. Lizardi. Next-generation sequencing-by-hybridization. *Nature Biotechnology*, 26:649–650, 2008.

[30] D. Loakes, D. M. Brown. 5-nitroindole as an universal base analogue. *Nucleic Acids Research*, 22:4039–4043, 1994.

[31] Madprime. DNA chemical structure. *http://en.wikipedia.org/wiki/File DNA_chemical_structure.svg*. Wikimedia Commons, 2007.

[32] R. E. Mardis. Next-generation DNA sequencing methods. *Annual Review of Genomics and Human Genetics*, 9:387–407, 2008.

[33] D. Margaritis, S. Skiena. Reconstructing strings from substrings in rounds. *Proceedings 36th Symposium on Foundation of Computer Science*, 6(2):237–252, 1995.

[34] M. Margulies, E. Michael, E. A. William, et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437:376–380, 2005.

[35] A. Maxam, W. Gilbert. A new method for sequencing DNA. *Proceedings of National Academy of Sciences of the USA*, 74(2):560–564, 1977.

[36] B. McCarthy, J. Holland. Denatured DNA as a direct template for in vitro protein synthesis. *Proceedings of National Academy of Sciences of the USA*, 54(3):880–886, 1965.

[37] A. Nikolakopoulos, H. Sarimveis. A metaheuristic approach for the sequencing by hybridization problem with positive and negative errors. *Engineering Applications of Artificial Intelligence*, 21:247–258, 2008.

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

[39] P. A. Pevzner. l-tuple DNA sequencing: computer analysis. *Journal of Biomolecular Structures Dynamics*, 7:63–73, 1989.

[40] P. A. Pevzner, R. J. Lipshutz. Towards DNA sequencing chips. *Proceedings of the 19th International Symposium on Mathematical Foundations of Computer Science*, pp:143–158, 1994.

[41] F. P. Preparata, A. Frieze, E. Upfal. Optimal reconstruction of a sequence from its probes. *Journal of Computational Biology*, 6:361–368, 1999.

[42] F. P. Preparata, A. M. Frieze, E. Upfal. *On the power of universal bases in sequencing by hybridization.* ACM, 1999.

[43] F. P. Preparata, E. Upfal. Sequencing by hybridization at the information-theory bound - an optimal algorithm. *Journal of Computational Biology*, 7:621–630, 2000.

[44] F. P. Preperata, J. Oliver. DNA sequencing by hybridization using semi-degeneratae bases. *Journal of Computational Biology*, 11:753–765, 2004.

[45] M. Radom, P. Formanowicz. Algorithms for sequencing by hybridization problems based on non-classical DNA chips. *Przegląd Elektrotechniczny*, 10:97–100, 2010.

[46] F. Sanger, A. R. Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology*, 94(3):441–448, 1975.

[47] F. Sanger, S. Nicklen, A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of National Academy of Sciences of the USA*, 74(12):5463–5467, 1975.

[48] S. Skiena, G. Sundaram. Reconstructing strings from substrings. *Journal of Computational Biology*, 2:333–353, 1995.

[49] S. Skiena, G. Sundaram. Sequencing by hybridization in few rounds. *Lecture Notes in Computer Science, Algorithms - ESA 2003*, 2832:506–516, 2003.

[50] K. Too, D. Loakes. *Universal Base Analogues and their Applications to Biotechnology.* Wiley-VCH Verlag GmbH and Co. KGaA, 2008.

[51] D. Tsur. *Optimal Probing Patterns for Sequencing by Hybridization*, wolumen 4175. Springer Berlin/Heidelberg, 2006.

[52] T. Uzawa, A. Yamagishi, T. Oshima. Polypeptide synthesis directed by DNA as a messenger in cell-free polypeptide synthesis by extreme thermophiles. *The Journal of Biochemistry*, 131(6):849–853, 2002.

[53] J. Watson, F. Crick. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

[54] D. A. Wheeler, M. Srinivasan, M. Egholm, et al. The complete genome of an individual by massively parallel DNA sequencing. *Nature*, 452(7189):872–876, 2008.

[55] J. H. Zhang, L. Y. Wu, X. S. Zhang. Reconstruction of DNA sequencing by hybridization. *Bioinformatics*, 19:14–21, 2003.

[56] J. H. Zhang, L. Y. Wu, Y. Y. Zhao, X. S. Zhang. An optimal approach to the reconstruction of positional DNA sequencing by hybridization with errors. *European Journal of Operational Research*, 182:413–427, 2006.

[57] P. Zhang, M. Egholm, N. Paul, M. Pingle, D. E. Bergstrom. Peptide nucleic acid-DNA duplexes containing the universal base 3-nitropyrrole. *Methods*, 23:132–140, 2001.

[58] P. Zhang, W. T. Johnson, D. Klewer, N. Paul, G. Hoops, V. J. Davisson, D. E. Bergstrom. Exploratory studies on azole carboxamides as nucleobase analogs: Thermal denaturation studies on oligodeoxyribonucleotide duplexes containing pyrrole-3-carboxamide. *Nucleid Acids Research*, 26(9):2208–2215, 1998.