Wersja 1.3, 24.10.2023

Drobna uwaga dla Bioinformatyki: tekst poniżej był napisany dla Informatyki, rok temu. Powycinałem najbardziej oczywiste oczywistości, ale jeżeli wciąż trafi się coś w stylu tłumaczenie jak jest zbudowane DNA, to przepraszam :)

Podsumowanie projektu do zrobienia (szczegóły były już dokładnie tłumaczone na laboratorium):

1) Praca w grupach dwuosobowych, należy zaimplementować algorytm(y) rozwiązujące problem sekwencjonowania przez hybrydyzację, w tym w skład projektu wchodzą:
    a. Generator instancji problemu generujący spektrum hybrydyzacji, potencjalnie z błędami pozytywnymi i negatywnymi (czy i ile – zależy od przeprowadzonego testu). Typowy rozmiar DNA n > 300nt oraz n < 700nt, rozmiar oligonukleotydów k >= 6 oraz k <= 10. Zależnie od przypadku.
    b. Generator rozwiązań początkowych, zwany przeze mnie w ramach laboratoriów „generatorem rozwiązań losowych", bo to niestety dobrze oddaje jego typową „jakość" pracy.
    c. Aby wyjść powyżej progu oceny 4.0, należy chociaż spróbować zaimplementować algorytm metaheurystyczny, który na podstawie rozwiązania/rozwiązań, które wyszły z algorytmu z punktu wcześniejszego stara się ulepszyć rekonstrukcję DNA z elementów spektrum z instancji.
    d. Sprawozdanie, na które składa się:
        i. Opis stworzonych algorytmów
        ii. Testy algorytmów, tj. ich parametrów, jeśli jakoweś w ogóle mają. W przypadku metaheurystyki jest tam ich zawsze sporo, należy wybrać 2-3 kluczowe i przetestować ich pływ na jakość otrzymywanego rozwiązania. Np. a przypadku ACO parametrem może być liczba mrówek, współczynnik parowania feromonów, itd. W przypadku algorytmu genetycznego: procentowe szanse na krzyżowanie i osobno mutacje, parametry związane z algorytmem (pomocniczym) selekcji rozwiązań, itd. Tabu Search: długość listy ruchów zakazanych, wielkość przeszukiwanego sąsiedztwa, itd.
        iii. Testy instancji problemu, np. wpływ % błędów hybrydyzacji danego typu na jakość rozwiązań wychodzących z algorytmu mającego już jakoś ustalone i niezmienne parametry wewnętrzne (np. z testów z poprzedniego podpunktu). Wpływ długości dna (dość trywialny i oczywisty w wynikach tekst), wpływ długości parametru k, itd.
2) Na potrzeby wykresów/tabel wyników najlepiej używać miary odległości między dwoma łańcuchami znaków, np. odległość Levenshteina. Ponieważ aby stworzyć dowolną instancję problemu potrzebujemy zacząć od stworzenia DNA, to jak już na podstawie spektrum z błędami (lub bez) nasze algorytmu jakiś łańcuch DNA odtworzą, wtedy można odwołać się do tego oryginalnego łańcucha i obiektywnie ocenić stopień różnicy między obydwoma – na tej podstawie mamy punkty pomiarowe do wykresów w sprawozdaniu.
3) SBH klasyczne, znamy długość DNA, długość oligonukleotydów czyli k, liczbę błędów negatywnych, pozytywnych, wierzchołek początkowy.
4) Testy w sprawozdaniu w ogólności można podzielić na dwa rodzaje. Pierwszy typ testów bada wpływ parametrów algorytmu na jakość rozwiązań, druga kategoria testów bada wpływ wartości zmiennych opisujących instancję problemu (np. liczba błędów) na trudność znalezienia dobrego rozwiązania.
    a. Na potrzeby testowania parametrów algorytmu musimy dysponować pewnym stałym, reprezentacyjnym zbiorem instancji testowych. Powinny być do siebie względnie podobne i w liczbie min. kilkudziesięciu. Sugeruję przyjąć dla nich jeden stały parametr k, np. k= 9 lub 10, oraz stałą, albo podobną długość DNA czyli n (np. między 500 a 700, ta druga wartość już raczej dla k=10). Różnić się zdecydowanie powinny liczbą i rodzajem błędów hybrydyzacji. Załóżmy totalnie dla przykładu poniżej, że mamy 60 takich instancji tworzących zbiór testowy.
        i. Przykład: chcemy przetestować liczbę mrówek w metaheurystyce ACO, tj. wpływ tej liczby na jakość rozwiązania. Zakładamy, że dla danych i stałych parametrów ACO, liczba mrówek i jej wpływ na jakość rozwiązania będzie badana dla wartości 50, 100, 150, 200 i 250.
        ii. Zbieranie danych do testu / wykresu wygląda następująco. Najpierw każdą z 60 instancji przeliczamy przez metaheurystykę ACO dla liczby mrówek równej 50 (a wszystkie inne zmienne rządzące działaniem ACO są ustawione na **jakieś** wartości i ich nie zmieniamy już w tym teście. Przez przeliczenie rozumiem tu uzyskanie jakieś rekonstrukcji DNA. Potem dla każdej instancji porównujemy to co wyszło z algorytmu z DNA oryginalnym dla danej

instancji (miara Levenshteina). Mając 60 instancji, po 60-ciu uruchomieniach algorytmów mamy 60 miar Levenshteina. Dodajemy je do siebie i dzielimy sumę przez 60. Wychodzi na średnia arytmetyczna jakości rozwiązania dla 60 instancji zbioru testowego i liczby mrówek równej 50.

iii. Powtarzamy punkt powyższy dla liczby mrówek 100, 150, 200 i 250. Dostajemy łącznie 5 punktów na wykres do sprawozdania (5, bo pierwszy punkt pomiarowy to oczywiście średnia miara jakości miar Levenshteina dla liczby mrówek równej 50 opisany powyżej).

iv. Powyższy schemat można powtórzyć dla dowolnego parametru meteheurystyki, przetestować należy 2-3 parametrów, nawet jak ich jest więcej w algorytmie.

v. Jeżeli nie dysponujemy metaheurystyką to trzeba kombinować :) i próbować znaleźć jakiekolwiek istotne parametry w naszym algorytmie losowym czy jak go tam nazwiemy. Poza problemem takim, że jest to o wiele prostszy algorytm (niż dowolna metaheurystyka) i może mieć o wiele mniej parametrów, procedura z grubsza wygląda tak samo.

b. Testowanie parametrów instancji problemu. Czyli druga klasa testów w sprawozdaniu. Należy minimum 2-3 parametry testować, np. n / k (idą w parze zazwyczaj), liczbę błędów takich czy innych. Opiszę typowy test badania wpływu błędów negatywnych.

i. Generujemy sobie zbiorek instancji testowych. Tutaj wartości n oraz k powinny być **identyczne**, załóżmy, że chcemy 20 instancji. Tworzymy więc 20 łańcuchów DNA, robimy z nich spektrum. **Musimy zapamiętać dokładnie to 20 wygenerowanych DNA** – czemu, o tym dalej, przydadzą się jeszcze później.

ii. Załóżmy, że chcemy przetestować wpływ błędów negatywnych dla wartości 2, 4, 6, 8 i 10% spektrum. Załóżmy też, że dla naszych 20 instancji testowych zawsze mamy DNA gdzie n = 600, k = 10. W ramach szukania pierwszego punktu na wykres do sprawozdania, w każdej instancji usuwamy 2% spektrum, czyli tworzymy błędy negatywne.

iii. Każdą z tych 20 instancji z błędami przeliczamy algorytmem takim jaki mamy (w algorytmie, czy to losowym czy metaheurystycznym wszystkie parametry traktujemy jako stałe i niezmienne w testach dalszych). Otrzymujemy 20 zrekonstruowanych łańcuchów DNA, każdy porównujemy z oryginalnym DNA dla danej instancji – wychodzi miara Levenshteina dla każdego. Dodajemy 20 wyników do siebie, sumę dzielimy przez 20. Otrzymana liczba to punkt na wykresie, na osi Y – wartość miary Levenshteina, na osi X oznaczamy „2% bł. negatywnych" lub po prostu „2%" bo tytuł wykresu będzie nam ogłaszać co dokładnie testujemy w danej chwili i co jest na osi OX.

iv. Teraz uwaga: dla 4% błędów negatywnych **musimy użyć tych samych 20 DNA które sobie zachowaliśmy na początku**. DNA, tj. jego struktura wewnętrzna, tj. kolejność i powtarzalność liter, ma wpływ na jakość wyników, a chcemy testować tylko wpływ błędów. Dlatego bierzemy te same DNA. Tym razem jednak w spektrum każdego z nich usuwamy 4% elementów. Po czym powtarzamy punkt trzeci tutaj i finalnie otrzymujemy teraz punkt wykresu oznaczający średnią jakość rekonstrukcji DNA według miary Levenshteina dla 4% błędów negatywnych.

v. Powtarzamy dla takich %-ów błędów negatywnych ile i jakie chcemy testować.

c. Testy dla ambitnych: używamy DNA z repozytoriów internetu. Geny mogą mieć „trudne" DNA do sekwencjonowania SBH, tj. trudniejsze niż to nasze radosne generowania łańcuchów tekstowych na alfabecie A, C, G, T z prawdopodobieństwem każdej literki 25%, ale to i tak nic w porównaniu z sekwencjami niekodującymi. Których w genomie jest zazwyczaj wielokrotnie więcej niż kodujących. W sekwencjach niekodujących (upraszczam) liczba powtórzeń pojedynczych liter może być znacznie większa niż przyjęty parametr k, co sprawi że dla takiego DNA algorytmy dla metody SBH będą leżeć i płakać. Albo raczej my patrząc na otrzymane wyniki, nawet wygenerowane przez niby „dobry" algorytm.

i. https://www.ncbi.nlm.nih.gov/nuccore/OQ859931.1?report=fasta
Powyżej przyjaciel wszystkich, koronawirus, wersja jedna z milionów zsekwencjonowanych. Tutaj w wersji DNA, pomimo że to wirus RNA. Tutaj dygresja: jeżeli traficie na sekwencje RNA, to będzie mieć U zamiast T, wystarczy podmienić.
https://www.ncbi.nlm.nih.gov/nuccore/OQ859931

to sekwencja białkowa wirusa. Wystarczy w prawej górnej części klikną link „FASTA” aby zobaczyć wersję DNA.

ii. Oczywiście te sekwencje zawsze są dłuższe niż kilkaset literek. Musimy po prostu sobie wyciąć z ich jakiś fragment o takiej długości jak nasze założenie co do parametru *n* dla danego testu.

iii. Genom człowieka: https://www.ncbi.nlm.nih.gov/genome/guide/human/

iv. Pseudo gen theta-defensyny: https://www.ncbi.nlm.nih.gov/gene?Db=gene&Cmd=DetailsSearch&Term=170949 Pseudo, bo już nie działa w naszym genomie. Zmutował nieszczęsny poliaminokwas dziesiątki tysięcy lat temu w genomie i to tak, że już nie da się w organizmie zbudować funkcjonalnego białka theta-defensyny. Mutacja trywialna. Jedna literka nukleotydu zmieniła się w inną. Tylko tak pechowo, że zamiast jak było wcześniej kodowania w tym miejscu genu jednej z 20 sekwencji aminokwasowych budujących białko, powstał tam tzw. kodon-STOP, który służy za stoper procesu produkcji genu na bazie sekwencji RNA/DNA. Czemu mikrobiologów ten fakt wkurzył? Białko wykazuje silne działanie anty retro-wirusowe. Retrowirusem jest HIV. Defensyna działa i jest wytwarzana przez komórki u niektórych małp naczelnych. Tak, dokładnie tych, które są odporne na działanie wirusa HIV, który nie może im wtedy rozwalić układu odpornościowego. Kurtyna :)