Uniwersytet Warszawski

Wydział Matematyki, Informatyki i Mechaniki

Aleksander Mućk

Nr albumu: 382184

Algorytmy do klastrowania duplikacji genomowych

Praca licencjacka na kierunku BIOINFORMATYKA I BIOLOGIA SYSTEMÓW

Praca wykonana pod kierunkiem dra hab. Pawła Góreckiego

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data

Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data

Podpis autora pracy

Streszczenie

Niniejsza praca przedstawia podstawowe propozycje rozwiązań algorytmicznych dla problemów klastrowania duplikacji genomowych w oparciu o scenariusze ewolucyjne. W części pierwszej wprowadzane są pojęcia dotyczące drzew genów, gatunków, modeli ich uzgadniania oraz tworzenia scenariuszy ewolucyjnych. Omówiony został również problem przeliczania i klastrowania duplikacji genomowych w scenariuszach ewolucyjnych. W części drugiej opisana została proponowana heurystyka wraz z przykładowymi testami oraz jej implementacja w języku Python.

Słowa kluczowe

duplikacja genu, drzewo genów, drzewo gatunków, analiza filogenetyczna, drzewo uzgadniające, Python, scenariusz ewolucyjny, strata genu, minimalizacja kosztu ewolucyjnego

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.9 Inne nauki matematyczne i informatyczne

Klasyfikacja tematyczna

Computional biology, Applied computing, Life and medical sciences

Tytuł pracy w języku angielskim

Algorithms for the clustering of genomic duplication

Spis Treści

| W | prowadzenie | Ę |
|----|--|----|
| 1. | Podstawowe pojęcia1.1. Drzewa genów i gatunków1.2. Uzgodnienie drzewa1.3. Modele scenariuszy drzew | 7 |
| 2. | Heurystyka | g |
| 3. | Dokumentacja użytkowa i opis implementacji | 11 |
| 4. | Podsumowanie | |
| Α. | Główna pętla programu zapisana w języku Python | 15 |
| В. | Przykładowe drzewo gatunków | 17 |
| С. | Przykładowe drzewa genów | 19 |
| D. | Przykładowy wynik działania programu (dla zbioru guigo) | 21 |
| Bi | bliografia | 23 |

Wprowadzenie

Analizy drzew filogenetycznych są, przez rozmiar danych i coraz bardziej skomplikowane modele, niezwykle złożone zarówno obliczeniowo jak i koncepcyjnie.

Drzewo genów zawiera geny. Drzewo gatunków zawiera gatunki.

Krótki opis zależności na bardzo podstawowym poziomie.

Praca składa się z czterech rozdziałów i dodatków. W rozdziałe 1 przedstawiono podstawowe pojęcia dotyczące drzew genów, drzew gatunków oraz modeli i scenariuszy ewolucyjnych. Rozdział 2 przedstawia propozycję heurystyki wraz z jej testami na rzeczywistych danych. W rozdziałe 3 opisano implementację i sposób użycia programu napisanego na podstawie przybliżonej we wcześniejszym rozdziałe heurystyki. Ostatni rozdział zawiera przemyślenia dotyczące możliwego użycia algorytmu i perspektyw jego rozwoju. W dodatkach umieszczono fragmenty kodu, przykładowe dane wejściowe i wyniki działania algorytmu.

Podstawowe pojęcia

Pojęciem pierwotnym dla scenatiuszy ewolucy

1.1. Drzewa genów i gatunków

Opis drzewa ogólny

Definicja 1.1.1 Opis.

Definicja 1.1.2 Opis.

1.2. Uzgodnienie drzewa

Podstawowy opis idei.

1.3. Modele scenariuszy drzew

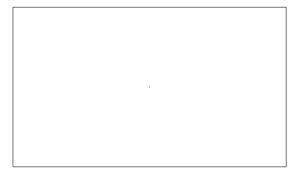
Podstawowy opis idei. Podział ruchów:

Definicja 1.3.1 Opis.

Definicja 1.3.2 Opis.

Dozwolone modele:

- 1. PG
- 2. FHS;



Rysunek 1.1: Obrazek scenariusza

Heurystyka

Dane na wejściu : drzewa genów $G_1...G_k$ dla każdego Gi jest mi scenariuszy opisanych jako wektory vi1 vi2 ... vim z epizodami (wg outputu ze skryptu)

Zróbmy wektor przybliżające rozwiązanie ME, nazwijmy je v*, które pasuje do każdego scenariusza. Wystarczy brać max po współrzędnych z każdego wektora v.... Oczywiście jest to rozwiązanie słabe bo kosztowne ale pasuje do każdego scenariusza.

Teraz w pętli spróbujmy poprawiać współrzędne z wektora v*: - obniż jedna z wartości w wektorze v - zaakceptuj w/w zmianę jeśli dla każdego drzewa genów istnieje scenariusz, który jest zgodny z takim wektorem epizodów Zakończ jeśli nie da się poprawić żadnej współrzędnej.

Wybór współrzędnej do obniżki: testować np. od końca wektora (od korzenia), od początku (od liści), losowo.

Dokumentacja użytkowa i opis implementacji

Opis działania programu.

Podsumowanie

W pracy przedstawiono pierwszą heurystykę.

Trudno przewidzieć wszystkie nowe możliwości, ale te co bardziej oczywiste można wskazać już teraz. Są to:

- opisy naszych algorytmów optymalizacyjnych,
- opisy naszych algorytmów optymalizacyjnych,
- $\bullet\,$ opisy naszych algorytmów optymalizacyjnych,

4.1. Perspektywy wykorzystania

DOPYTAĆ

Dodatek A

Główna pętla programu zapisana w języku Python

```
max_trees = []
        for scenario in self:
            all_dup_pref = [tree.duplication_prefix for tree n scenario]
            max_trees.append(self.rate_scenario(all_dup_pref))
        max_tree = self.rate_scenario(max_trees)
        if select_type == "random":
            index_list = [x for x in range(len(max_tree)) if x != 0]
            while index_list:
                index_list_position = random.randint(0, len(index_list) - 1)
                index = index_list[index_list_position]
                max_tree_temp = max_tree[:]
                max_tree_temp[index] -= 1
                for scenario in self:
                    for tree in scenario:
                        for i in range(len(tree.duplication_prefix)):
                            if max_tree_temp[i] - tree.duplication_prefix[i] < 0:</pre>
                                break
                        else:
                            break
                        index_list.pop(index_list_position)
                        break
                else:
                    max_tree = max_tree_temp
            return max_tree, sum(max_tree)
```

Dodatek B

Przykładowe drzewo gatunków

 $(\operatorname{prot},(\operatorname{fung},((\operatorname{chlo},\operatorname{embr}),(\operatorname{arth},((\operatorname{acoe},\operatorname{anne}),(\operatorname{echi},(\operatorname{chon},(\operatorname{oste},(\operatorname{amph},(\operatorname{moll},((\operatorname{mamm},(\operatorname{aves},\operatorname{rept})),\operatorname{agna}))))))))))$

Dodatek C

Przykładowe drzewa genów

```
(((aves,mamm),amph),chon)
  (((amph,aves),mamm),chon)
  ((((acoe,mamm),chlo),fung),prot)
  ((((echi,arth),mamm),embr),fung),prot)
  ((((echi,arth),mamm),prot),embr)
```

Dodatek D

Przykładowy wynik działania programu (dla zbioru guigo)

```
-FHS-
Data loaded. 0.0total random
Done in 2.3286948204040527.
start
Done in 0.0676581859588623.
end
Done in 0.24111151695251465.
index random
Done in 0.0971217155456543.
   —PG-
Data loaded. 0.0total random
Done in 0.18230891227722168.
start
Done in 0.006891012191772461.
Done in 0.006652355194091797.
index random
Done in 0.007452487945556641.
```

Bibliografia

[Zen69] Zenon Zenon, *Użyteczne heurystyki w analizie*, Młody Technik, nr 11, 1969.