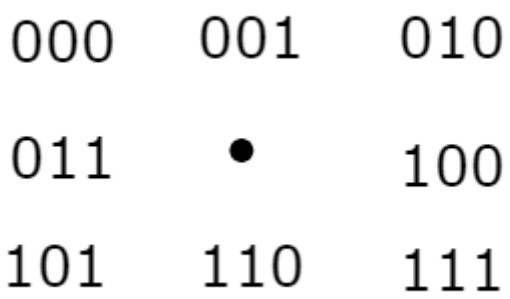
GENETSKI ALGORITEM ZA POZICIONIRANJE LABEL V RAZTRESNEM GRAFIKONU

Luka Bajić

Ta dokument opisuje genetski algoritem za pozicioniranje label v raztresenem grafikonu. Na raztresenem grafikonu je množica točk, vsaki točki pripada ena labela (npr. besedilo ali številka). Namen tega algoritma je, da postavi labele na takšen način da se ne prekrivajo in da jih je čimveč vidnih. Gre za optimizacijski problem, torej ni nujno da algoritem najde optimalno rešitev. Cilj je, da najde čimkvalitetnejšo rešitev v sprejemljivem času.

Algoritem je sestavljen iz naslednjih podkorakov:

- Inicializacija:
  - Vsaka labela ima 8 možnih lokacij relativno na točko: zgoraj levo, zgoraj center, ... in ima odmik 5 pikslov v vsako smer, kot je razvidno na spodnji sliki.



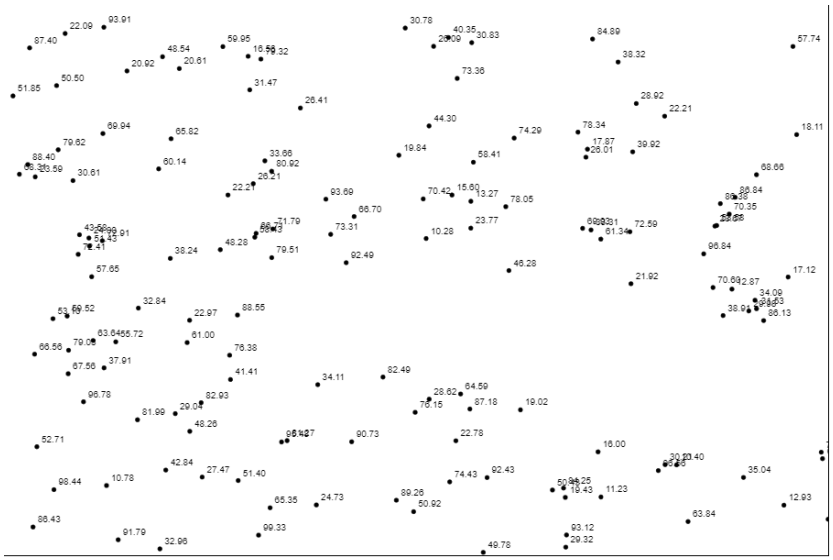
  - Kot je razvidno iz slike je vsaka možna lokacija labele predstavljena s tremi biti
  - En osebek je sestavljen iz 3 bitov za vsako točko na grafikonu. Na začetku je vsak osebek sestavljen iz naključnih bitov.

  - Ena generacija je sestavljena iz N osebkov (v mojem algoritmu je N = 100, lahko pa je poljubna vrednost, odvisno od tega koliko časa imamo na voljo za procesiranje in koliko kvalitetno rešitev želimo poiskati.

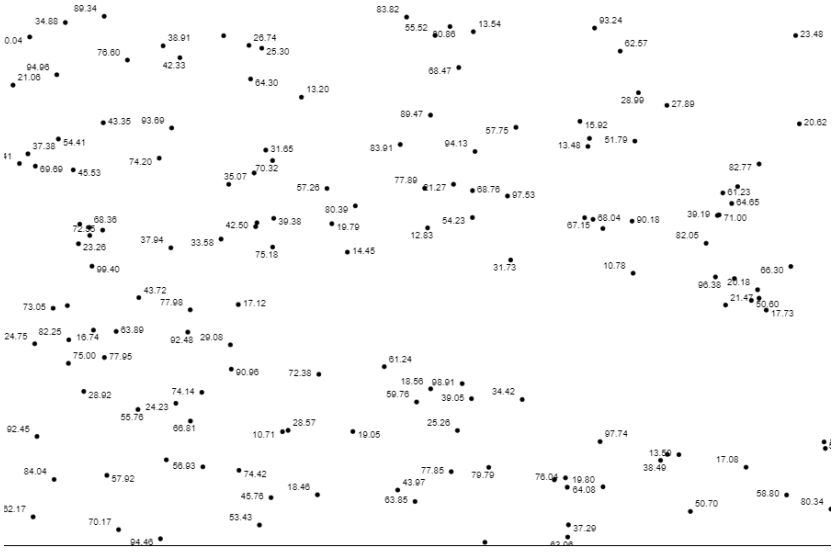
- Selekcija:
  - Najprej je potrebno določiti najboljših k osebkov iz generacije, na podlagi katerih se nato generirajo naslednje generacije (v mojem algoritmu sem izbral k = 15). Najboljše osebke določimo z uporabo funkcije kvalitete.
  - Funkcija kvalitete je funkcija, ki pove koliko dober je nek osebek. V mojem algoritmu sem kot funkcijo kvalitete izbral število prikazanih label, ker je to glavni cilj tega algoritma. Lahko bi upoštevali tudi npr. število prikazanih robnih label oziroma ekstremnih vrednosti, saj so le te običajno pomembnejše za človeka, ki analizira grafikon.

- Križanje in mutacija:
  - Ključni element genetskega algoritma je križanje. Namen križanja je, da iz trenutnih osebkov generira boljše osebke (torej rešitev, ki je bližja optimalni). Iz osebkov, ki so bili izbrani pri selekciji se torej generira nova generacija z N osebki.
  - V mojem algoritmu sem križanje implementiral tako, da se vzamejo naključni pari in vsak osebek v paru prispeva polovico svoje genetske kode. Koda novega osebka je torej sestavljena iz polovice kod vsakega izmed staršev. V določenih problemskih domenah dosežemo boljše rezultate tudi s križanjem več kot dveh osebkov hkarti. Prav tako je včasih smiselno, da ne oba osebka prispevata enak delež kode, temveč npr. naključen delež ali pa boljši osebek prispeva večji delež.
  - Mutacija pa služi temu, da vnese nekaj naključnosti v križanje z upanjem, da algoritem najde neko lokalno optimalno rešitev, ki je samo s križanjem ne bi našel. V mojem algoritmu sem mutacijo implementiral tako, da v vsaki generaciji spremeni en naključen bit naključnega osebka. Odvisno od problemske domene in podatkov bi lahko tudi povečali ali zmanjšali pogostost mutacije. Večja verjetnost mutacije je smiselna predvsem v bolj naključnih podatkih, medtem ko v urejenih podatkih to ni tako nujno potrebno.
  - Spodnja slika prikazuje delovanje algoritma. Na zgornji sliki je začetno stanje z privzeto nastavitvijo lokacije labele zgoraj desno, na spodnji pa stanje po koncu izvajanja algoritma. Kot vidimo prekrivanja točk na spodnji sliki ni več, ker jih algoritem ne dopušča, so pa zaradi tega določene labele izpuščene.

- Zaustavitev algoritma:
  - Za zaustavitev sem v mojem algoritmu uporabil fiksno število generacij, ker je hitrost izvajanja dokaj dobra.



(a) Pred izvajanjem



(b) Po izvajanju

– Če imamo opravka z grafikoni, ki imajo večje število točk oz. bolj zahtevnimi problemi je smiselen način zaustavitve tudi časovna omejitev.
– Če pa želimo čimkvalitetnejšo rešitev, lahko uporabimo tudi zaustavitev, ko funkcija kakovosti preseže neko določeno vrednost.