Универзитет у Нишу

Електронски факултет Ниш

Интелигентни системи

Први домаћи задатак – Генетски алгоритми

Студент:

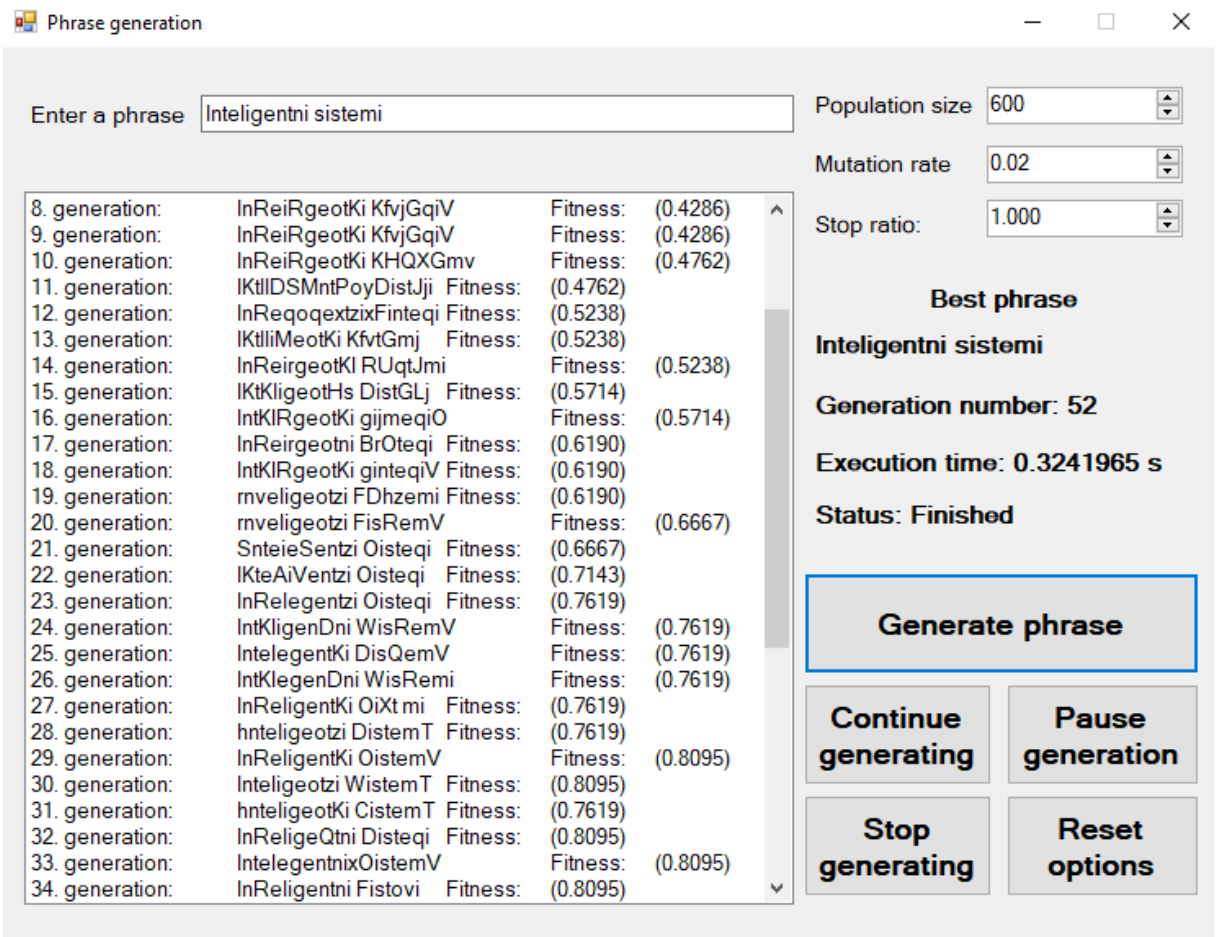
Димитрије Јовић, 928

Ниш, децембар 2019. год.

С А Д Р Ж А Ј

1. ФОРМУЛАЦИЈА ПРОБЛЕМА

Хладно децембарско вече, као и већина људи у Србији, Мирко са својим оцем Јанком прати свој омиљени квиз. Како то обично и бива, током трајање квиза уједно траје и такмичење између оца и сина ко ће знати више одговора. Међутим, приликом игре у којој је потребно погодити тачан редослед фигура, дошло је до мале опкладе. Наиме, Мирков отац, Јанко, је тврдио да је само човек у стању да погоди тачну комбинацију фигура у задатом временском периоду. Знајући да његов отац слабо познаје област рачунарства, а пошто му је био потребан новац, Мирко је предложио да ако он успе да реализује такву апликацију добије жељену количину новца. Након десетак минута убеђивања и сугерисања, договор је постигнут. Договор је гласио да уколико Миркова апликација рекреира реченицу, односно неку фразу, коју зада његов отац Јанко да он добије жељену суму новца. Након пар дана, видевши да је применом алгоритма грубе силе, брзина генерисања спора, решио је да се обрати свом другу Миладину, како би убрзао генерисање. Током састанка, дошли су до закључка да би се применом генетских алгоритама побољшала ефикасност апликације, након чега је Мирко кренуо у имплементацију. После неколико дана, Мирко је успео да заврши своју апликацију, која је са успехом рекреирала сваку фразу, односно реченицу коју је задао његов отац.



2. ИМПЛЕМЕНТАЦИЈА ГЕНЕТСКОГ АЛГОРИТМА

Сама имплементација генетског алгоритма је рађена од нуле. У оквиру имплентације постоје две класе: *Phrase* и *Population*. Што се тиче најутицајнијих параматера, то су величина популације и вероватноћа мутације.

2.1. Phrase класа

Класа *Phrase* представља такозвани *DNK* наше јединке. Класа садржи два атрибута:

- низ карактера – сваки карактер представља такозвани хромозом у оквиру DNK ланца, хромозоми су кодирани по вредности, где сваки хромозом може бити или карактер (било које велико или мало слово) или бланко знак.
- фитнес – представља меру доброте *DNK* у односу на тражени *DNK*. Изражен је у интервалу између нуле и јединице.

У оквиру ове класе, имплементиране су основне операције над хромозомима, односно операција мутације и укрштања (*crossover*).

Што се тиче операције укрштања, односно *crossover*-а, имплементиран је *crossover* са једном тачком укрштања. Одабир места, односно тачке укрштања је случајан, за сваки пар родитеља. Ако је за неки пар родитеља тачка укрштања била на шестом хромозому, за следећи пар родитеља та тачка неће бити иста. По принципу случајности се такође бира и распоред хромозома у новој јединки, односно да ли ће прво ићи хромозоми једног, односно другог родитеља.

Операција мутације у зависности од коефицијента мутације, врши мутирање, односно замену хромозома неком случајном вредношћу из опсега вредности дефинисаних у оквиру задатка.

Поред основних операција над хромозомима, имплементирана је и функција доброте. То је веома једноставна функција, имплементирана на начин да пролази кроз *DNK* и броји колико је хромозома на својим позицијама, На крају се тај број дели са укупном дужином хромозома, тако да се добија вредност између нуле и јединице.

2.2. Population класа

Као што јој и само име каже, класа *Population* представља популацију над којом се врше основне генетске операције. Од битнијих атрибута, класа садржи:

- *mutationRate* – представља коефицијент мутације, односно вероватноћу да ће се мутација над хромозомом десити. Одабир коефицијента врши корисник, и може бити у интервалу од 0 до 1.
- *selectionList* – представља листу родитеља базирану на њиховој доброти која ће се користити приликом селекције родитеља. Елементи листе су објекти класе *Phrase*.
- *population* – представља тренутну популацију, односно низ објеката класе *Phrase*.

Од функција ћемо издвојити функцију која врши генерисање нове популације као и функцију за генерисање *selectionList*.

Функција за генерисање *selectionList*-е ради на начин да се прво проналази јединка са највећом добротом у популацији, након чега се на основу доброте изабране јединке, врши израчунавање колико пута је потребно додати одређену јединку у листу. Тако да ће се након завршетка израчунавања, у листи највећи број пута наћи она јединка која има највећу доброту, односно највећу вероватноћу избора.

Што се тиче генерисања нове популације, врши се случајна селекција родитеља из *selectionList*-е, након чега се примењују основне операције над јединком, односно укрштање и мутација, и врши додавање новонастале јединке у популацију. Величину популације бира корисник и креће се у интервалу између сто и две хиљаде јединки.

Што се тиче извршења самог алгоритма, сам алгоритам се извршава дотле, док не задовољи услов за излазак. Тај услов представља доброту коју најбоља јединка мора да задовољи. Вредност доброте коју најбоља јединка мора да испуни бира корисник, а вредност се креће у интервалу између нуле и јединице.