Математички факултет

Универзитет у Београду

**СЕМИНАРСКИ РАД**

У оквиру курса Рачунарска интелигенција

Тема:

*Хибридизација Генетског и*

*VNS Алгоритма*

Професор: Асистенти: Студенти:

Александар Картељ Стефан Капунац Марко Савић,149/2019

Денис Аличић Мирко Кордић,242/2019

**Април, 2023.**

**САДРЖАЈ**

**1.Увод**

**1.1** Предмет истраживања

1.2 Пример проблема

1.3.Потребни програми и ресурси

2.Учитавање и обрада полазних података

3.Генетски алгоритам

3.1.Селекција

3.2.Укрштање

3.3.Мутација

4.VNS

5.Хибридизација

6.Анализа резултата

7.Закључак

8.Литература

**1.Увод**

* 1. **Предмет истраживања**

У овом семинарском раду, приказан је рад различитих комбинација генетског алгоритма и VNS алгоритма.

Идеја је да се на конкретном проблему испробају како појединачни алгоритми тако и њихове различите хибридизације, те да се потом упореде њихови резултати.

* 1. **Пример проблема**

За пример проблема на којем би рад ових алгоритама био тестиран узет је “Проблем трговачког путника” (TSP) , јер у неку руку представља изазов за решавање коришћењем ових алгоритама, због заснованости на пермутацијама о којима посебно треба водити рачуна приликом примене генетског алгоритма.

Проблем трговачког путника (Travelling salesman problem) представља проблем у коме је потребно да за задату листу градова и међусобних растојања међу њима, одредимо редослед обиласка градова тако да рута којом се крећемо буде најкраћа могућа, при чему сваки град треба да буде посећен тачно једанпут.

Конкретан скуп података на коме су тестирани алгоритми представља низ градова Западне Сахаре који се састоји од 29 градова задатих својим координатама.

**1.3. Потребни програми и ресурси**

Комплетан код писан је у програмском језику Python, а извршен је у окружењу Jupyter Notebook. Коришћена је python бибилиотека *networkx* и њене имплементације рада са графовима,библиотека *matplotlib* за цртање графика, и друге основне python библиотеке. Рачунар на ком су тестирани алгоритми садржи 8GB RAM меморије.

**2.Учитавање и обрада полазних података**

Улазни подаци представљају низ од 29 градова Западне Сахаре, задати својим координатама. Зарад компактнијег чувања података и пропратних информација које су потребне током израчунавања, одлучено је да се подаци о градовима чувају у оквиру графа. У те сврхе искоришћена је python библиотека *networkx,* која омогућава ефикасан рад над графовима. Подаци су сачувани у оквиру потпуно повезаног графа, чији су чворови градови, задати својим *x* и *y* координатама, док су гране одређене паром чворова које спајају и додатном иформацијом о њиховом растојању. За рачунање растојања између градова коришћено је еуклидско растојање.

**3. Генетски алгоритам**

Генетски алгоритам је један од еволутивних алгоритама инспирисаних Дарвиновом теоријом еволуције. Припада стохастичним алгоритмима претраге са хеуристиком. Најчешће се користи за проблеме у дискретном домену али може се генерализовати коришћењем одговарајућих пресликавања.

Генетски алгоритми су засновани на посматрању популације јединки и њиховом мењању кроз генерације.

Одлучено је да се јединке представљају њиховим кодом који је представљен пермутацијом бројева од 0 до 28. Друга обавезна компонента сваке јединке јесте фитнес функција, којим се мери квалитет јединке. Фитнес ће у нашем конкретном примеру представљати дужину пута представљеног кодом јединке. Рачунање се изводи једноставним пролажењем кроз код јединке и сабирањем растојања између суседних градова(чворова).

Како генетски алгоритми спадају у еволутивне алгоритме, који врше оптимизацију коришћењем техника инспирисаних природном еволуцијом као што су селекција,укрштање и мутација, тако ћемо проћи кроз сваки од ових концепата и размотрити њихов начин имплементације у нашем конкретном примеру.

У свим имплементацијама коришћен је елитизам, још једна битна карактеристика еволутивних алгоритама, која спречава губљење добрих јединки. Иако ће се најбоље јединке најверованије налазити у скупу одабраних родитеља, применом укрштања и мутације може доћи до тога да потомци имају лошији квалитет. Приступ подразумева да се неколико најбољих јединки одабере и директно прекопирају у наредну генерацију, при чему број елитних једники не сме бити превелики.

**3.1. Селекција**

За процес селекције одабрана је класична турнирска селекција, јер њена својства умањују нам омгућавају да донекле лакше балансирамо између експолатације и експлорације.

Као параметре функције селекција добија популацију јединки, величину турнира и листу забрањених јединки. Листа забрањених јединки се прослеђује како не би дошло укрштања истих родитељских јединки.

На почетку се прави листа дозвољених индекса уклањањем забрањених из листе индекса свих јединки популације.

Потом се из листе дозвољених индекса случајним избором бира tournament\_size јединки које ће учествовати у процесу селекције.

Једноставним проласком кроз одабране јединке вршимо претрагу оне са најмањим фитнесом. Таква јединка ће бити резултат процеса селекције.

**3.2. Укрштање**

За процес укрштања одабрано је једнопозиционо укрштање, где се из дужине кода јединке случајним избором бира *breakpoint* на основу ког вршимо генерисање деце од кодова родитељских јединки.

Проблем код позиционог укрштања може да настане код пермутација, које су основ нашег примера. Ако бисмо позиционон укрстили могли бисмо да добијемо невалидне конфигурације деце (понављање неког броја). Због тога је одабрана модификација позиционог укрштања под називом *укрштање првог реда*.

Оно се састоји из наредних корака:

1. Изаберемо регион у првом родитељу
2. Пресликамо изабрани регион у прво дете на исто место
3. Десно од тог региона(циклично) постављамо гене из другог родитеља. Ако се ген који желимо да ставимо већ налази у детету прескачемо га, и посматрамо наредни ген.
4. Аналогно радимо и за друго дете

**3.3 Мутација**

Како је наш проблем заснован на пермутацијама, тако је и процес мутације морао бити прилагођен томе.

Како би се одржала форма пермутације кода јединке за процес мутације одабрана је ротација два различита гена кода.

Битан параметар функције мутације јесте вероватноћа мутације, о којој ће бити речи касније.

**4.VNS**

Променљиво околинско претраживање (Variable Neighborhood Search - VNS) је метахеуристички оптимизациони алгоритам који има за циљ да пронађе најбоље решење датог оптимизационог проблема у простору свих могућих решења. Он је дизајниран да се може применити на различите оптимизационе проблеме, укључујући комбинаторне, континуалне и мешовите проблеме.

VNS алгоритам почиње са почетним решењем и наставља истраживање простора решења итеративним примењивањем различитих околинских структура на тренутно решење. Околинска структура дефинише скуп суседних решења која се могу добити прављењем малих измена на тренутном решењу.

На свакој итерацији, алгоритам оцењује решења у околини тренутног и прихвата најбоље као ново тренутно решење. Ако најбоље решење у околини тренутног није боље од тренутног, алгоритам се премешта на другу околинску структуру и понавља процес.

Алгоритам наставља да истражује простор решења итеративним примењивањем различитих околинских структура, сједињавајући их у један оптимизациони поступак. Он наставља са претрагом свих околинских структура до испуњења услова заустављања, као што је достигнуће максималног броја итерација или одређеног нивоа квалитета решења.

VNS је показао ефикасност у решавању различитих оптимизационих проблема, укључујући транспорт, логистику и инжењерство, и показао се као моћна и ефикасна техника оптимизације.

За потребе овог рада коришћене су две врсте VNS алгоритма. Основни нередуковани VNS који након сваког креирања *new\_solution* позива локалну претрагу како би покушао да унапреди новокреирано решење, и редуковани VNS који овај корак прескаче.

**5.Хибридизација**

Хибридизација генетског алгоритма (ГА) и Variable Neighborhood Search (VNS) је метод оптимизације која комбинује две различите технике у циљу постизања бољих перформанси у решавању различитих оптимизационих проблема.

Хибридизацијом ГА и VNS -а могу се уједно користити предности оба приступа. ГА и VNS имају комплементарне квалитете. ГА је добро прилагођен за истраживање великог простора претраге и налажење разноликог скупа решења, док је VNS добар у интензивирању претраге и истраживању околина око обећавајућих решења, тј. може се користити за детаљнију локалну претрагу у мањим регионима простора претраге. Ово може омогућити ефикасније и прецизније проналажење глобалних и локалних оптимума у различитим врстама оптимизационих проблема.

Хибридизација ГА и VNS-а може имати значајне предности у поређењу са појединачном применом ГА или VNS-а, посебно за проблеме који имају комплексну структуру претраге и велики број променљивих које треба оптимизовати. Хибридизација ГА и VNS-а може омогућити ефикасније и прецизније проналажење глобалних и локалних оптимума у различитим врстама проблема.

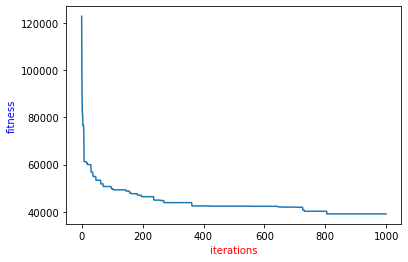
**6.Aнализа решења**

У овом поглављу ћемо приказати резултате примене различитих варијанти наведених техника над описаним проблемом.

Оптимално решење проблема је 27603.

* **VNS**

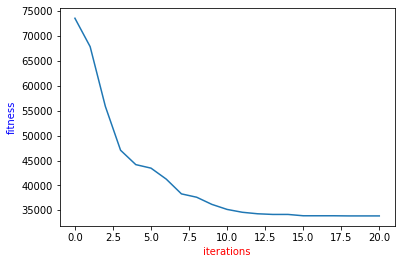
Први примењен алгоритам био је VNS. Креирана је насумична јединка, и над њом је покренут VNS алгоритам са 10 околина, 1000 итерација и вероватноћом преласка у другу околину 0.5.



Као резултат добија се 39059, што је далеко од оптималног. То можемо приметити и са графика промене фитнеса кроз итерације, где фитнес доста споро конвергира ка решењу јер тек у 800-тој итерацији долази до стагнирања решења. Такође може се приметити да је потребан велики број итерација да би се поправило неко осредње решење, док лошија решења попраља у малом броју итерација.

* **Генетски алгоритам**

Покренут је генетски алгоритам са 20 итерација, величином популације 1000, коришћеним елитизмом величине 10, вероватноћом мутације 0.15 и величином турнира 10.



Овај алгоритам ће надаље бити коришћен као „бенчмарк“ алгоритам

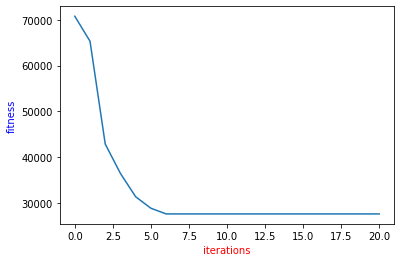
Добијен је резултат од 33874. Видимо да генетски алгоритам брзо конвергира решењу у односу на VNS. Након 15. итерације долази до локалном оптимуму из ког не успева да изађе до краја извршавања. Можемо приметити да је решење које добијамо значајно боље и да се добија у знатно мањем броју итерација.

* **Хибрид генетског алгоритма и редукованог VNS -а**

Овај хибрид подразумева да се редуковани VNS алгоритам примењује над сваком јединком, а потом над најбољом јединком генерације још једном.

RVNS се позива са 2 околине,5 итерација и вероватноћом преласка у другу околину 0.5 . Параметри су смањени због брзине извршавања и незнатне разлике у коначном решењу.

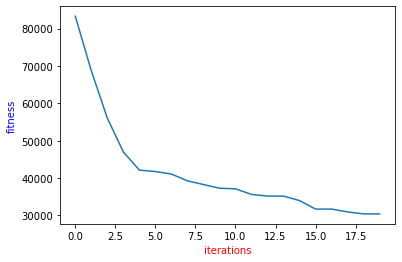
Параметри генетског алгоритма су остали исти као у претходној анализи.



Добијено решење оваквог алгоритма је 27603. Можемо приметити да овакав алгоритам још брже конвергира свом оптимуму него обичан генетски алгоритам,такође и да је функција зависности фитнеса од броја итерације доста глађа од функције код генетског алгоритма. Видимо да је овај алгоритам успео да пронађе оптимaлно решење проблема.

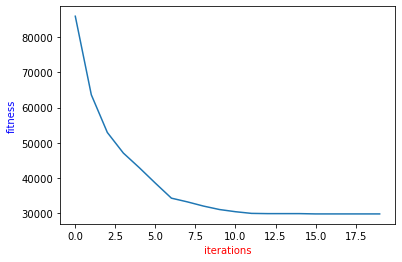
* **Хибрид генетског алгоритма и VNS -а**

Овде је примењен нередуковани VNS на сваку јединку у генерацији. Првобитно је генетски алгоритам покренут са стандардним параметрима : итерације = 20, величина популације = 1000, итд. Међутим због велике сложености алгоритма, који позива VNS над сваком јединком, где и VNS позива локалну претрагу, време извршавања овог алгоритма је доста велико те је одлучено да се смањи величина популације на 100.



Добијено решење је 30416, што је обзиром да је број итерација остао исти (20) ,а популација је смањена са 1000 на 100 добро решење. Оно што можемо приметити јесте да конвергенција није готова у потпуности те да не би било лоше повећати број итерација. Међутим покретањем алгоритма са бројем итерација 30, долази до стабилизације графика, али се решење не побољшава.

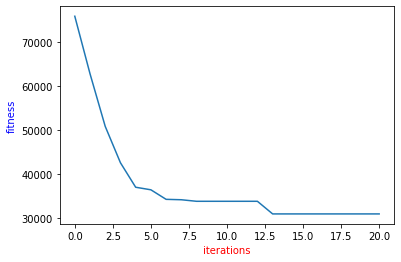
Како повећање броја итерација није помогло, испробано је и повећање популације, са 100 на 150.



Добије је решење од 29739. Видимо да је повећањем популације дошло до побољшања решења, али такође и доста брже конвергенције, што је и очекивано узевши и обизир горе наведене особине самог VNS-а.

* **Хибрид генетског алгоритма и VNS -а који се примењује само на најбољу јединку у генерацији**

У овој варијанти хибридизације користи се нередуковани VNS алгоритам, који се примењује искључиво на најбољу јединку у генерацији. Ова варијанта је испробана због бржег извршавања алгоритма. Параметри VNS -а су : околине = 7, итерације = 30, вероватноћа преласка = 0.5 , док су параметри генетског алгоритма остали исти.



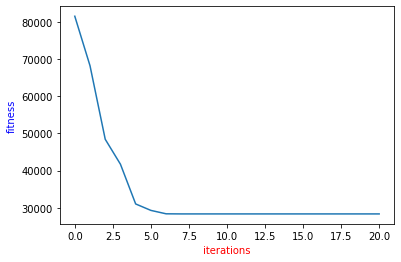
Резултат примене овог алгоритма је 30899. Након неколико покретања овог алгоритма закључили смо да често даје лошије резултате као и да је конвергенција спорија у односу на остале хибридне варијанте. Ово је такође очекивано узевши у обзир особине VNS-а.

* **Хибрид генетског алгоритма и VNS -а који се примењује на најбољу јединку након краја генетског алгоритма**

Овај вид хибрида подразумева да се генетски алгоритам извршава применом редукованог VNS -а над сваком јединком, а затим се пронађено решење генетског алгоритма покушава унапредити применом нередукованог VNS -а над њим.

За овај тип хибрида смо се определили због знатно мање сложености редукованог VNS -а од нередукованог, како би се позив алгоритма брже завршио.

Како се нередуковани VNS у овом случају извршава само једанпут, повећали смо параметре : околине: 20, итерације: 200.

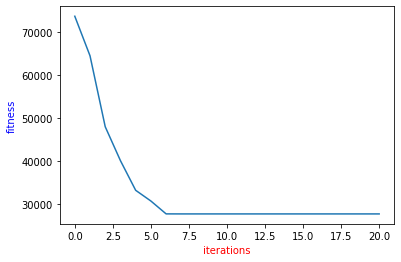


Добијено решење је 28318. Можемо приметити да је график доста сличан графику хибридизације где се редуковани VNS примењује над сваком јединком. То је очекиван резултат јер VNS који је покренут над најбољом пронађеном јединком није успео да поправи њен фитнес.

* **Хибрид генетског алгоритма и VNS -а са динамички променљивом вероватноћом мутације**

Циљ ове хибридизације био је да се вероватноћа мутације у оквиру генетског алгоритма динамички мења у зависности од њеног фитнеса. Идеја иза тога је била да јединке са лошијим фитнесом имају већу вероватноћу мутације не би ли дошло до њиховог побољшања. Формула која је коришћена је специфична за конкретан пример, добијена је експерименталним путем и гласи : mutation\_prob - 4000/new\_population[i].fitness

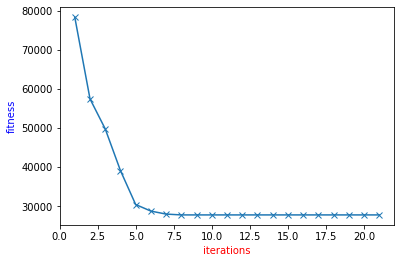
Што се тиче VNS -а коришћена је комбинација, где се редуковани VNS примењује над сваком јединком, а нередуковани над најбољим у генерацији. Параметри су остали исти.



Добијено је решење од 27748.

* **Хибрид генетског алгоритма и VNS-а са динамички променљивом вероватноћом мутације везаном са бројем итерација**

Код овог хибрида редуковани VNS се примењује над сваком јединком, потом се над сваком најбољом у генерацији примењује нередуковани VNS (оваква комбинација је направљена због смањења сложености). Оно што карактерише овај хибрид јесте динамички променљива мутација која зависи од броја итерације. Овај хибрид настао је вођен идејом да су јединке у почетним генерацијама лошије него у каснијим итерацијама и да тада вероватноћа мутације треба бити највећа, те да се кроз итерације она смањује. За вероватноћу мутације коришћена је формула 1/(2\*ј).



Добијено је решење од 27748.

**7. Закључак**

Као што је наведено у опису техника, VNS доста повећава експлоатацију и убрзава конвергенцију ка решењу. То се може и видети из тога што сви изведени хибриди ова два алгоритма брже теже ка пронађеном решењу од самог генетског алгоритма.

Међутим, као и код обичног генетског алгоритма у већини хибриднихних случајева долази до заглављивања у локалном оптимуму. Један од разлога за то јесте што сам VNS тешко успева да поправи фитнес добрих јединки. То смо могли приметити у случају када је RVNS примењиван над сваком јединком генерације, а VNS само над јединком која је представљала коначан резултат ГА. Његова улога се сводила на то да јединке поправи у почетним итерацијама и самим тим убрза конвергенцију ка неком оптимуму.

Примена нередукованог VNS-а над сваком јединком у великој популацији није била могућа због велике сложености, зато смо се определили да тај проблем превазиђемо смањењем параметара алгоритма и различитим комбинацијама примене VNS-а у оквиру ГА. Тиме смо добијали релативно добра решења, нека су брже нека спорије конвергирала али сва су се заглављивала у локалним оптимумима.

Вођени идејом да је популација у почетним генерацијама разноврсна и да кроз генерације фитнеси јединки постају све бољи следеће решење је било да вероватноћу мутације прилагодимо тој чињеници. У реализацији те идеје имали смо две имплементације, међутим ниједна не даје значајне промене у резултатима.

Разматрана је и идеја праћења фитнеса те у генерацијама када нема побољшања да се увећа вероватноћа мутације, не би ли на тај начин успели да изађемо из локалних оптимума. Од ове идеје смо брзо одустали, јер алгоритам у тренутку када пронађе свој оптимум, због стагнирања фитнеса, почиње да повећава вероватноћу мутације чиме нарушава квалитет популације што није много различито од тога да се само покрене поново алгоритам и исчекује боље решење.

Можемо приметити да хибрид ГА и RVNS даје веома добре резултате. Ова варијанта је мање сложености од нередукованог VNS-а стога се брже извршава и омогућава примену и над великим популацијама за разлику од VNS-а.

Како смо већ напоменули оптимално решење проблема је 27603. Оптимално решење је у приказаном покретању алгоритама достигнуто коришћењем хибрида генетског алгоритма и редукованог VNS алгоритма. Међутим велики број наведених хибридних решења је током писања овог рада достизао оптимално решење.

Наш финални закључак је да хибрид VNS-а и ГА-а значајно убрзава конвертенцију ка оптимумy (углавном локалном), али са друге стране превише експлоатише добра решења.

**8. Литература**

1. „Computational Intelligence - An Introduction“, Andries Engelbrecht, John Willey & Sons, 2007
2. Материјали са курса Рачунарска интелигенција