Универзитет у Бањој Луци I Архитектонско-грађевинско-геодетски факултет
University of Banja Luka I Faculty of architecture, civil engineering and geodesy
ISSN 2303-6036 **UDK** I UDC 72

[4] 2016 4[1]

АГГ+ часопис за архитектуру, грађевинарство, геодезију и сродне научне области
ACEG+ Journal for Architecture, Civil Engineering, Geodesy and other related scientific fields

036-055 **Оригинални научни рад** I Original scientific paper
UDK I UDC 69:519.857
DOI 10.7251/AGGPLUS1604048C
COBISS.RS-ID 6666776
Рад примљен I Paper received 29/07/2016
Рад прихваћен I Paper accepted 28/09/2016

Горан Ћировић
Висока грађевинско-геодетска школа Београд, Хајдук Станка 2, Београд, Србија, cirovic@sezampro.rs

Драган Николић
Висока грађевинско-геодетска школа Београд, Хајдук Станка 2, Београд, Србија, nikolic@vggs.rs

Снежана Митровић
Висока грађевинско-геодетска школа Београд, Хајдук Станка 2, Београд, Србија, mitrozs@sezampro.rs

ПРИМЕНА МЕТОДА МЕКОГ ПРОГРАМИРАЊА У ГРАЂЕВИНАРСТВУ

APPLYING THE SOFT COMPUTING METHODS IN CIVIL ENGINEERING

Оригинални научни рад
Original scientific paper
Рад прихваћен I Paper accepted 28/09/2016
UDK I UDC 69:519.857
DOI 10.7251/AGGPLUS1604048C
COBISS.RS-ID 6666776

Горан Ћировић
Висока грађевинско-геодетска школа Београд, Хајдук Станка 2, Београд, Србија, cirovic@sezampro.rs

Драган Николић
Висока грађевинско-геодетска школа Београд, Хајдук Станка 2, Београд, Србија, nikolic@vggs.rs

Снежана Митровић
Висока грађевинско-геодетска школа Београд, Хајдук Станка 2, Београд, Србија, mitrozs@sezampro.rs

ПРИМЕНА МЕТОДА МЕКОГ ПРОГРАМИРАЊА У ГРАЂЕВИНАРСТВУ

АПСТРАКТ

Поступак проналажења оптималног решења представља процес добијања најбољег резултата у датим околностима. Инжењери свакодневно доносе мноштво одлука било да обављају неку од менаџерских функција или се баве пројектовањем, изградњом или одржавањем објеката. Крајњи циљ доношења свих тих одлука је остваривање додатне добити проналажењем минимума или максимума циљне функције којом је моделиран проблем. У раду су описане три методе меког програмирања (теорија грубих скупова, генетски алгоритми и куку претрага). У анализи одлучивања оне омогућавају коришћење више различитих врста алата и техника које се користе у процесу стварања модела за доношење одлука. То се пре свега односи на разјашњења, тумачења и предузимања акција у циљу повећања кохезије између могућности одређене ситуацијом и циљева и вредновања система од стране укључених експерата или доносилаца одлуке. У раду је приказан и пример примене грубих скупова при избору оптималне локације за фабрику бетона.

Кључне речи: инжењерска оптимизација, меко програмирање, груби скупови, генетски алгоритми

APPLYING THE SOFT COMPUTING METHODS IN CIVIL ENGINEERING

ABSTRACT

The process of finding the optimal solution is the process of obtaining the best result under the given circumstances. Engineers continuously bring a multitude of decisions irrespective of whether they are engaged in the managerial functions or in the design, construction and maintenance of facilities. The ultimate goal of making all of these decisions is to get additional profit by finding the minimum or maximum of fitness function that models the problem. This paper describes three methods of soft programming (Theory of Rough Sets, Genetic Algorithms and Cuckoo Search) that allow making use of many different types of tools and techniques in the analysis used in the process of creating a model for decision making. This primarily relates to the clarification, interpretation, and taking actions to increase cohesion between the capabilities determined by a particular situation and objectives and evaluation of system used by the involved experts or decision makers. This paper also presents the example of the rough sets application in selecting optimal locations for a concrete factory.

Key words: engineering optimization, soft programming, rough sets, genetic algorithms

1. УВОД

Операциона истраживања представљају грану математике која се бави применом научних метода и техника помоћу којих се предлаже најбоље или задовољавајуће решење у анализи одлучивања. Зачеци развоја метода оптимизације могу се приметити још у радовима Њутна, Лагранжа и Кошија. Развој диференцијалних рачунских метода оптимизације потпомогнут је достигнућима Њутна и Лајбница у области математичке анализе. Лагранж је такође познат по развоју метода оптимизације које поред ограничења садрже и Лагранжове мултипликаторе.

Упркос наведеним раним доприносима у области операционих истраживања, врло мали напредак је направљен до средине двадесетог века, када је напредак у развоју рачунара омогућио спровођење процедура оптимизације и стимулисао даља истраживања о новим методама. Развој рачунарства такође је утицао на масовну појаву литературе о техникама оптимизације и настанак неколико нових значајних области у операционим истраживањима [1].

Већина практичних проблема у свакодневној пракси које је потребно разматрати првобитно је описана на нејасан или непрецизан начин. Ово се пре свега односи на одређивање одговарајућих циљева, ограничења о томе шта може да се уради, међусобних релација између параметара који се проучавају и других утицаја изван система, могућности примене алтернативних решења и рокова за доношење одлука. Процес дефинисања проблема је од кључног значаја, јер у великој мери утиче на то какви ће релевантни закључци проистећи из истраживања. Тешко је издвојити "тачан" одговор на "погрешно" постављен проблем!

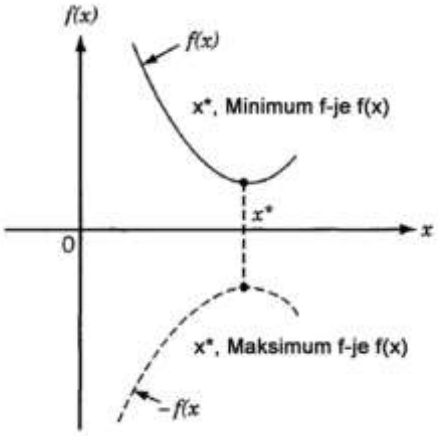
На прикупљању релевантних података о проблему обично се проведе изненађујуће велика количина времена. Већина података је потребна да би се добило тачно разумевање проблема и да би се обезбедиле потребне улазне вредности за математички модел који треба формулисати у следећој фази истраживања. Често већи део потребних података није доступан када се започине са истраживањем, било зато што се информације нису никада адекватно складиштиле или зато што су застареле, или су у погрешном формату. Потребна је помоћ више кључних људи у организацији да би се пратили сви витални подаци. Чак и са овим напорима, већи део података може бити врло непоуздан, односно представља грубе процене само на основу емпиријског искуства. Обично, доста времена се проводи покушавањем да се побољша прецизност података и тиме се покуша добити што тачније решење.

Општи приступ решавању проблема може се дефинисати кроз варијантна решења за специфичне проблеме и неопходан је методичан приступ, применом алгоритама или “корак-по-корак” процедура помоћу којих се долази до решења. У најједноставнијем случају то значи да је потребно наћи минимум или максимум дефинисане функције проблема и систематски направити избор променљивих у оквиру дозвољених интервала вредности (Слика 1).

Пре почетка рада на имплементацији метода за решење проблема неопходно је да проблем буде комплетно дефинисан и да се отклоне и најмање сумње на шта се заправо он односи. На основу свега горе наведеног, оптимизација се може описати као поступак добијања најбољег резултата у датим околностима[1].

Током пројектовања, извођења и одржавања грађевинских објеката, инжењери су принуђени да доносе одлуке у свакој од фаза реализације пројекта, при чему оне могу

значајно утицати на на коначну вредност инвестиције. Крајњи циљ доносиоца одлука је смањење утрошка потребних ресурса, односно повећање жељене добити и у било којој ситуацији се може изразити у функцији променљивих параметара. У том случају, оптимизација се дефинише као процес проналажења максималне или минималне вредности функције за дефинисане параметре, а технике којима се то постиже обједињене су у посебну научну дисциплину названу операциона истраживања.



Слика 1. *Произвољна циљна функција f(x) [2]*

Пошто се са сигурношћу може тврдити да је проблем коректно дефинисан, следећа фаза се односи на формулацију проблема у облику који је погодан за анализу. Конвенционални приступ за то је изградња математичког модела који разматра суштину проблема. Пре него што се покрене дискусија о томе како да се формулише такав модел, потребно је истражити природу модела и математичког модела засебно.

Математички модели су идеализовано представљени проблеми који се изражавају преко математичких симбола и израза. Математички модел оптимизације у грађевинарству може се приказати системом једначина и сродних математичких израза који описују суштину проблема. Тако да, ако постоји n међусобно мерљивих одлука које је потребno донети, оне се могу представити као решење променљивих одлучивања (нпр. k1, k2,..., kn) чије је одговарајуће вредности потребно утврдити.

Мерене перформансе у једном систему (нпр. добит или цена производа) се изражавају као математичка функција променљивих одлучивања (на пример, P=3*k1+2*k2+…+kn) и називају се функције циља. Ограничења која се могу доделити променљивим одлучивањима су такође математички изрази приказани најчешће кроз неједнакости (на пример, x1-3x1*x3 <10) и дефинишу се као услови ограничења. Коефицијенти са десне стране у условима ограничења и функцији циља називају се параметри модела. Проблем оптимизације се може дефинисати математичким моделом уз услов да се изаберу вредности променљивих одлучивања, како би се пронашао минимум или максимум функције циља у складу са наведеним условима ограничењима. Наведени модел са варијацијама током формулисања проблема симболизује модел који се користи у операционим истраживањима. Одређивање одговарајућих вредности параметрима модела (једна вредност за један параметар) је критични део процеса реализације модела

и захтева истраживање и прикупљање релевантних података. Математички модели имају много предности у односу на лингвистички описане проблеме.

Једна од предности је да математички модел описује проблем много концизније – чини тенденцију да укупна структура проблема буде разумљивија. То помаже да се открију важни узрочно-последични односи између параметара којима се описује модел. Добар приступ у развоју модела је да се отпочне са развојем једноставнијих модела, а затим треба разматрати сложеније моделе који више одражавају сложеност разматраног проблема. Овај процес надградње модела се наставља док год модел остаје прилагодљив, односно док год се повећава његова прецизност.

Развој компликованих математички модела је на неки начин истоветан развоју озбиљних рачунарских програма. Када се прва верзија рачунарског програма заврши, неизбежно је да садржи много грешака. Такав програм мора бити темељно тестиран да би се покушало што је могуће више пронаћи и исправити грешака. На крају, после дугог низа побољшања програма, потребно је да програмер (или тим програмера) закључује да актуелна верзија програма генерално даје валидне резултате. Иако је могуће да неке мање грешке несумњиво остану скривене у програму (и можда никада не буду откривене), главне грешке су елиминисане чиме се програм може поуздано користити.

Слично томе, прва верзија математичког модела неминовно садржи многе недостатке. Неки од релевантних фактора, или међусобно битни односи често нису укључени у модел, као што и неки од битних параметара нису правилно процењени. То је неизбежно, с обзиром на тешкоће комуникације и разумевања свих аспеката и потешкоћа приликом прикупљања поузданих података. Зато, пре него што модел буде први пут коришћен, мора бити темељно тестиран. На крају, после дугог низа побољшања модела закључује се да актуелни модел даје разумно валидне резултате. Иако неки мањи недостаци такође могу остати скривени, главне недостатке је потребно елиминисати да се модел може поуздано користити.

2. ПРИМЕНА ТЕОРИЈЕ ГРУБИХ СКУПОВА

Третирање проблема неизвесности, нејасности и непрецизности је једна од кључних активности у успешној реализацији интелигентних система за подршку у одлучивању. До сада су развијени бројни приступи за решавање тог проблема, а један од најновијих математичких прилаза пружа теорија грубих скупова. Базира се на истраживањима групе аутора на челу са проф. Павлаком, спроведеним на Варшавском Универзитету, на Институту за компјутерске науке крајем 80-тих година прошлог века [3].

Елементарни скуп је било који скуп свих објеката који се не разликују и представљају грануле знања о универзуму. Оштар (прецизан) скуп је било која унија неких елементарних скупова, а у супротном случају скуп је груб. Претпоставка да објекти могу да се “виде” само кроз доступне информације о њима, доводи до становишта да је структура знања грануларна. Због грануларности знања неки објекти који нас интересују не могу се препознати и појављују се као исти или слични. Као последица тога, неодређени појмови, супротно прецизним појмовима, не могу се окарактерисати у смислу информација о њиховим елементима.

Приступ теорије грубих скупова овај проблем решава претпоставком да се било који неодређени објекат може описати паром одређених појмова који се називају доња и

горња апроксимација (Слика 2). Доња и горња апроксимација су две основне операције у теорији грубих скупова. Доња апроксимација се састоји из свих објеката који сигурно припадају скупу, а горња апроксимација садржи све објекте који му вероватно припадају (они који сигурно припадају и они за које се не може са сигурношћу тврдити да припадају). Разлика између горње и доње апроксимације чини гранично подручје неодређеног појма (објекта) [4].

У циљу математичке формулације грубих скупова полази се од табеле података. Подразумева се употреба појма атрибут уместо појма критеријум, јер је први појам знатно уопштенији од другог. Као табела података подразумева се четворострука група података:

$$S = (U, A, V, f) \tag{1}$$

где су:

U – коначан скуп објеката

A – коначан скуп атрибута

Уз сваки атрибут $a \in A$ придружен је скуп B а његових вредности или процене. Сваки атрибут a детерминише функцију $f_a: U \to V_a$. Уз сваки подскуп B од A, придружује се релација неразликовања I на U, означена као I(B) и сходно томе дефинисана као:

$$I(B) = \{(x, y) \in U \times U : f_a(x) = fb(y), \forall a \in B\} \tag{2}$$

Нека је У коначан скуп објеката – универзум и нека постоји X такво да је $X \subseteq U$, при чему $x \in X$. Уводи се бинарна релација B на U, односно релација неразликовања. Нека је B подскуп од A.

Дефинишу се следеће операције на скуповима:

$B*(X)$ - доња апроксимација од X дефинисана је како следи:

$$B*X = \{x \in U : B(x) \subseteq U\} \tag{3}$$

$B*(X)$ - горња апроксимација од X дефинисана је како следи:

$$B*X = \{x \in U : B(x) \cap X \neq 0\} \tag{4}$$

Гранично подручје X је скуп

$$BN_B(X) = B*(X) - B*(X) \tag{5}$$

Ако је гранично подручје X празан скуп:

$$BN_B(X) = 0 \tag{6}$$

скуп је оштар у односу на B, а у супротном случају

$$BN_B(X) \neq 0 \tag{7}$$

скуп X је груб у односу на B.

Према томе, доња апроксимација скупа је унија свих гранула знања које су потпуно укључене, односно садржане у скупу. Горња апроксимација је унија свих гранула које имају не-празан пресек са скупом. Гранично подручје је разлика између горње и доње апроксимације.

Слика 2. *Грануле знања, скупови и апроксимације*

Могу се дефинисати четири основне класе грубих скупова, односно четири категорије непрецизности:

$B*(X) \neq 0 \; i \; B*(X) \neq U, \; akko \; je$ X грубо B – одређена

$B*(X) \neq 0 \; i \; B*(X) \neq U, \; akko \; je$ X унутрашње B – неодређена

$B*(X) \neq 0 \; i \; B*(X) = U, \; akko \; je$ X спољашње B – одређена

$B*(X) \neq 0 \; i \; B*(X) = U, \; akko \; je$ X тотално Б – неодређена

У теорији грубих скупова постоје елементи (објекти) универзума који се не могу са сигурношћу сврстати у елементе неког одређеног скупа, и који чине груби скуп. Због тога, да би се проблем несигурности дефинисао, треба увести функцију припадности елемената грубом скупу, која се назива функција грубе припадности.

$$\alpha_B(X) = \frac{|B*(X)|}{|B*(X)|} \tag{8}$$

при чему $0 \leq \alpha_B(X) \leq 1$

где је $|X|$ кардиналност скупа X, $X \neq 0$. Коефицијент $\alpha_B(X)$ је тачност апроксимације појма X.

Уколико је:

$\alpha_B(X) = 1$ скуп је оштар у односу на B, а уколико је:

$\alpha_B(X) < 1$ скуп је груб у односу на B

Функција грубе припадности објекта x грубом скупу се дефинише на следећи начин

$$\mu_X^B(x) = \frac{\left|X \bigcap B(x)\right|}{\left|B(X)\right|} \tag{9}$$

при чему је $0 \le \mu_X^B(x) \le 1$

3. ПРИМЕНА ЕВОЛУЦИОНИХ АЛГОРИТАМА У ИНЖЕЊЕРСКИМ ОПТИМИЗАЦИЈАМА

Спектакуларни напредак у области операционих истраживања потпомогнут је развојем достигнућа у области рачунарства и свеобухватних истраживања примене техника оптимизације. Алгоритми инспирисани природом се развијају у XX веку налазећи примену у науци, економији, техници и тако даље. Заснивају се на неким од основних принципа природе и социјалног понашања. Генетски алгоритми, користећи принципе неодарвинистичке еволуције, развијају генерације јединки које воде ка оптималном решењу [5].

С развојем рачунарства и повећањем процесне моћи рачунара, људи су почели решавати изузетно комплексне проблеме који су до тада били нерешиви – и то напросто техником исцрпне претраге , односно претраживањем целокупног простора решења. С обзиром на то да су данас рачунари екстремно брзи, није ретка ситуација да се њиховом употребом у једној секунди могу претражити милиони или чак милијарде потенцијалних решења, и тиме врло брзо пронаћи најприхватљивије решење. Овакав приступ је оправдан само ако решавамо проблем који се исцрпном претрагом може решити. Нажалост, постоји читав низ проблема који не спадају у ову категорију.

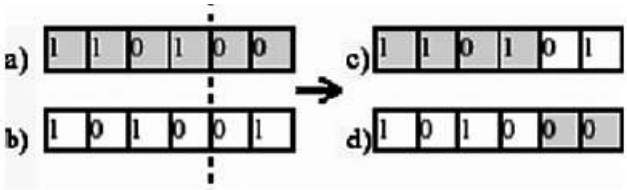
На срећу, оптимално решење често није нужно и обично је прихватљиво и решење које је довољно добро. Алгоритми који проналазе решења која су задовољавајућа или довољно добра, али не нуде никакве гаранције да ће успети пронаћи оптимално решење називају се приближни алгоритми или хеуристичке методе. У савременим истраживањима веома су заступљене метахеуристике, као скуп алгоритамских концепата који се користе за дефинисање хеуристичких метода применљивих на широк скуп проблема. Може се рећи да је метахеуристика хеуристика опште намене чији је задатак усмеравање проблемски специфичних хеуристика према подручју у простору решења у којима се налазе добра решења. Примери метахеуристика су симулирано каљење, табу претраживање, еволуцијско програмирање итд. [6].

Еволуцијско програмирање данас се дели на три велика подручја: генетске алгоритме, еволуцијске стратегије и еволуцијско програмирање. Заједничка им је идеја да раде с популацијом решења над којима се примењују еволуцијски оператори (селекција, укрштање, мутација, замена), чиме популација из генерације у генерацију постаје све боља и боља.

3.1. ГЕНЕТСКИ АЛГОРИТМИ

Генетски алгоритми представљају најпопуларнију технику претраге засновану на еволуцијском рачунарству. Традиционални генетски алгоритми се исказују низовима бита фиксне дужине. Претпоставља се да свака позиција у низу означава одређену карактеристику јединке, а додељена вредност утиче на коначну вредност функције циља. Низ карактеристика који се најчешће придружује јединки представља низ независних параметара, као аналогију у односу на гене у биолошким организмима. Сваки ген представља ентитет који је структурално независан од других гена.

Генетски материјал се у биолошким организмима преноси укрштањем, стога се сматра једним од главних оператора репродукција у генетским алгоритмима. На основу два низа битова као низа хромозома родитеља формирају се нове јединке од делова хромозома родитеља. Мутација је такође битан оператор репродукције у којој један бит мења место у низу другим битом и формира нови генетски материјал потомства (Слика 3).



Слика 3. *Графички приказ укрштања хромозома родитеља (а,б) и хромозома потомства (ц,д)*

Могуће је идентификовати два приступа решавању одређеног проблема: прилагодити проблем генетском алгоритму или генетски алгоритам прилагодити специфичностима проблема. Уколико се одабере прилагођавање алгоритма проблему, потребно је модификовати његов рад тако да разматрани параметри буду величине својствене одређеном задатку. Најчешће је реч о употреби или дефинисању другачијих структура података и оператора. За велики број таквих случајева развијени су специјализовани генетски алгоритми, који се у том случају називају еволуцијским програмима. Тиме се постижу значајна повећања делотворности и такви производи се могу наћи на софтверском тржишту и имају велику предност у примени.

Овакав начин захтева доста рада на прилагођавању. Зато се компромис постиже прављењем еволуцијског програма који ће моћи да се примени на читаву класу проблема. С друге стране, ако се настоји решити проблем уз помоћ генетског алгоритма, тада је потребно проблем прилагодити генетском алгоритму. Могуће решење се најчешће приказује као низ битова (бинарни приказ). Међутим, постоји читав низ проблема за које је тешко или немогуће применити бинарни приказ. Уобичајни генетски оператори могу генерисати значајан проценат немогућих решења која не доносе никаква побољшања, већ се само успорава алгоритам [7].

Функција циља или функција доброте јединке се у литератури још назива фитнес функција или функција оцене квалитета и представља израз:

$$dobrota(v)=f(x),$$

где бинарни вектор v представља реалан број x [dg, gg], а dg, gg $\in R$. Што је доброта јединке већа, јединка има већу вероватноћу укрштања. Функција циља је кључ за процес

селекције. За задати проблем оптимизације највећу потешкоћу представља дефинисање функције циља, која би требало да одражава суштину проблема који се решава.

3.1.1. Једноставни генетски алгоритми

Многим техникама претраге неопходни су помоћни параметри или информације да би се метода успешно имплементирала (градијентним техникама претраге неопходни су изводи да би се добио тражени максимум или минимум), док генетски алгоритми не захтевају те помоћне информације. ГА безусловно претражују простор трагајући за што бољим решењем, а за што је неопходна само функција циља са дефинисаним односом између параметара који се вреднују том функцијом. ГА користе правила транзиције вероватноће да усмере претрагу простора према што прихватљивијем решењу.

Холанд је предложио (једноставни) генетски алгоритам као рачунарски процес који имитира еволуцијски процес у природи и примењује га на апстрактне јединке. Сваки еволуцијски програм одражава популацију јединки у некој одређеној генерацији, док свака јединка представља потенцијално решење проблема који се обрађује. Свака јединка је представљена једнаком структуром података (број, низ, матрица, стабло итд.) [8].

Структура и начин функционисања једноставних генетских алгоритама (simple genetic algorithm – SGA) је релативно једноставан и своди се на копирање и замену делова хромозома. Применом SGA добијају се добри резултати претраге у различитим областима инжењерства и састоје се од три операције: репродукције, укрштања и мутације.

У природи, при свакој репродукцији долази до рекомбинације гена која узрокује различитост између јединки исте врсте, али и сличност с родитељима јединки. У генетским алгоритмима измена гена која настаје при репродукцији назива се укрштање, иако то није сасвим у складу са биологијом. Поред укрштања, уочава се још једна појава, али у знатно мањем опсегу. Реч је о случајном мењању генетског материјала који настаје под деловањем спољашњих фактора и назива се мутација. Укрштање и мутација се у генетским алгоритмима називају генетским операторима, а процес издвајања најспособнијих јединки у оквиру сваке генерације назива се селекција.

Пример SGA алгоритма [9]:

```
Генериши случајну популацију
Селектуј јединке за репродукцију (користећи функцију циља)
метод селекције:
-једноставна селекција
-турнирска селекција
-елиминацијска селекција
Укрштање хромозома родитеља
Мутирање хромозома потомака
Додај потомство назад у популацију
Елитизам
(Селектуј јединке за репродукцију)
```

SGA су погодни и ефикасни у случајевима:

- када је домен претраге широк, комплексан и недовољно дефинисан
- када сазнања о параметрима претраге нису довољна или када експертским системима није могуће смањити простор претраге
- када проблем није могуће анализирати математичким моделима

3.1.2 Прилагодљиви генетски алгоритми

Прилагодљиви генетски алгоритми (AGA – Adaptive Genetic Algorithm) представљају групу ГА чији параметри, као што су величина популације, односно вероватноћа укрштања и мутације је променљива током извршавања ГА. Прилагодљиви ГА су предложени 1994. године у раду [10]. Да би се вероватноћа укрштања и мутације правовремено мењала, потребно је на неки начин одредити тренутно стање популације ГА, да ли конвергира ка неком оптимуму.

Најједноставнији пример може се представити променом стопе мутације у складу са променама у популацији. Уколико се дужи временски период популација не побољшава, повећава се стопа мутације.

Процедура имплементације АГА може се дефинисати на следећи начин[10]:

корак I: Иницијална популација

Користи се величина популација добијена генерисањем случајног броја

корак II: Генетски оператори

Селекција – елитистичка стратегија у проширеном простору одабира јединки

Укрштање – оператор укрштања заснован на приоритетима укрштања

Мутација – оператор мутације заснован на активној локалној претрази

корак III: Примена локалне претраге помоћу методе успона на врх (hill climbing method) у ГА петљи

корак IV: Примена хеуристичких метода за прилагођавање параметара ГА (вероватноћа укршања и мутације)

корак V: Услови завршетка

Начелно се може рећи да ако је достигнут претходно дефинисани максимални број генерација или одређено оптимално решење током претраге, претрага се зауставља. У супротном се алгоритам поново враћа на други корак.

3.1.2. Приказ генетских оператора

Гени носе основне инструкције за изградњу генетских алгоритама. Сами хромозоми представљају низ гена којима се описује решење проблема претраге. Гени у ГА репрезентују вредност појединачних фактора унутар фактора контроле коме се додељује доња и горња граница. Овај опсег се може поделити на одређени број интервала који се представљају низом бита.

У начелу, хромозом може бити било каква структура података која описује својства једне јединке. За генетски алгоритам је значајно да хромозом представља могуће решење задатог проблема. За сваку структуру података потребно је дефинисати генетске операторе, док они опет треба да буду тако дефинисани да не стварају нове јединке које би представљале немогућа решења.

Приказ решења може битно утицати на учинак генетског алгоритма, стога је избор приказа изузетно значајан. Већина теорија које се односе на генетске алгоритме везане су управо за бинарни приказ решења. У пракси се показало да бинарни приказ даје најбоље резултате у већини примера и погодан је због своје једноставности у имплементацији. Процес избора две јединке из популације које ће укрштањем пренети генетски материјал на потомство назива се селекција.

Након избора начина кодирања, следећи корак се односи на начин избора јединки у популацији које ће стварати потомство као и број потомака који ће се генерисати. Сврха *селекције* је чување и преношење добрих својстава на следећу генерацију јединки. Селекцијом се бирају добре јединке које ће суделовати у следећем кораку репродукције. На тај начин се добар генетски материјал чува и преноси на следећу популацију, а лош одумире. Поступак селекције би се могао остварити сортирањем и избором најбољих јединки, као и умањењем популације за одређен број јединки, али такав поступак довео би до преране конвергенције генетског алгоритма [7].

Процес оптимизације би се практично завршавао у свега неколико првих итерација. Проблем је у томе што се овим поступком изгуби добар генетски материјал који могу имати лоше јединке. Зато је потребно осигурати и лошим јединкама да имају неку (мању) вероватноћу преживљавања. С друге стране, боље јединке треба да имају већу вероватноћу опстанка, односно већу вероватноћу учешћа у процесу репродукције.

Генетски алгоритми, с обзиром на врсту селекције, деле се на генерацијске и елиминационе. Генерацијски генетски алгоритам у једној итерацији располаже с две популације (што је уједно и недостатак генерацијског ГА), јер при избору добре јединке из старе популације које чине нову популацију и након селекције суделују у процесу репродукције.

Карактеристичне врсте селекција које користи генерацијски ГА су: једноставна селекција и турнирска селекција. С друге стране елиминацијска селекција је карактеристика елиминацијског генетског алгоритма (ГА with steady-state reproduction).

У процесу *укрштања* (crossover) суделују две јединке које се називају родитељи. Укрштање тиме представља бинарни оператор при чему настају једна или две нове јединке које представљају потомке. Најважнија карактеристика укрштања је да потомци наслеђују својства својих родитеља. Ако су родитељи добри (прошли су процес селекције), тада ће највероватније и потомство бити добро, ако не и боље од својих родитеља.

Укрштање може бити дефинисано са произвољним бројем прекидних тачака. Униформно укрштање је укрштање са b-1 прекидних тачака (b је број битова). Вероватноћа да потомак наследи својство једног родитеља је 0.5, односно једнака је вероватноћи наслеђивања својстава за оба родитеља. Ако се те вероватноће разликују за поједине гене, тада се такво униформно укрштање назива p-униформно укрштање.

На пример, ако је p=0.3, тада је вероватноћа да ће један бит бити наслеђен од првог родитеља 30%, а од другог 70%. Ако је вероватноћа наслеђивања различита за поједине гене, тада се задаје маска која дефинише за сваки ген посебно која је вероватноћа наслеђивања.

У зависности од тога да ли замена битова код оператора укрштања зависи од позиције или броја замењених битова, дефинише се позициона и дистрибуциона склоност оператора укрштања. Укрштање са једном тачком прекида има максималну позициону, а

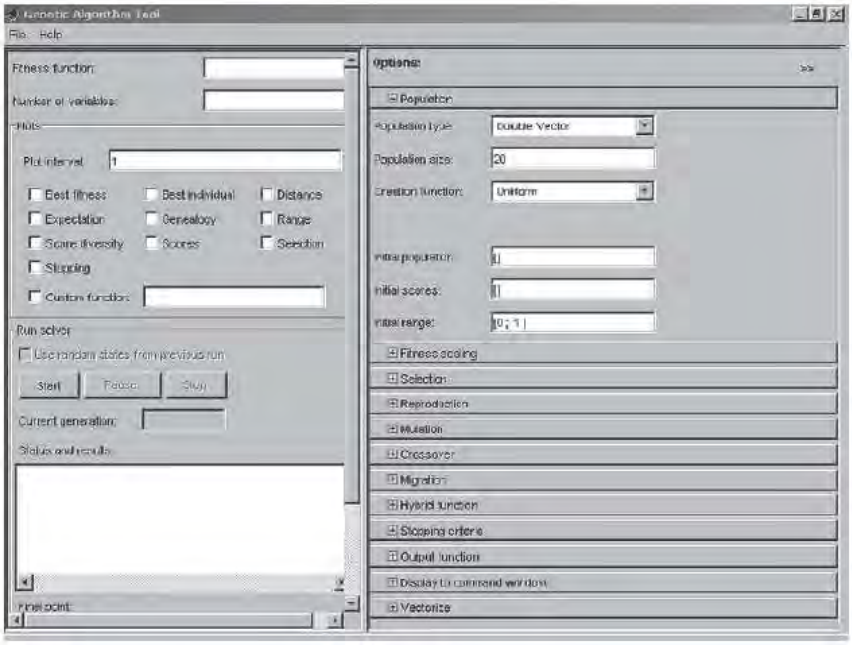
минималну дистрибуциону склоност. Вероватноћа замене неког бита у хромозому зависи искључиво од позиције тог бита. Друга крајност је униформно укрштање које има минималну позициону склоност, а максималну дистрибуциону, јер сви битови се замењују без обзира на њихову позицију у хромозому.

Укрштање с једном и две тачке прекида чува шему укрштања. Кад популација постане хомогена, простор који претражује алгоритам се смањује. Управо је то разлог зашто се то укрштање користи при већим популацијама, јер већа популација има и већу разноликост шема. Друга крајност је униформно укрштање које с великом вероватноћом ломи шеме, али претражује већи простор. Стога се униформно укрштање користи при мањим популацијама.

Други оператор који је карактеристичан за генетски алгоритам је мутација или случајна промена једног или више гена. *Мутација* је оператор који делује над само једном јединком. Резултат мутације је измењена јединка. Параметар који одређује вероватноћу мутације пм једног бита је уједно и параметар алгоритма. Ако вероватноћа мутације тежи ка јединици, тада се алгоритам претвара у алгоритам случајне претраге простора решења. С друге стране, ако вероватноћа мутације тежи ка нули, поступак ће највероватније већ у почетку процеса оптимизације стати у неком локалном оптимуму.

Једноставна мутација сваки бит хромозома мења с једнаком вероватноћом пм. Потпуна мутација случајним поступком бира хромозом (а не ген) за мутацију и тада свим битовима мења место. Мутацијом се претражује простор решења и управо је мутација механизам за избегавање локалних минимума током претраге простора. Ако укупна популација заврши у неком од локалних минимума, једино случајним претраживањем простора решења проналази се боље решење. Довољно је да једна јединка (настала мутацијом) буде боља од осталих, па да се у неколико следећих генерација све јединке преселе у простор где се налази боље решење.

Улога мутације се огледа и у обнављању изгубљеног генетског материјала. Уколико се догоди да све јединке популације имају исти ген на одређеном месту у хромозому, само укрштањем се тај ген никад не би могао променити. Ако је реч о бинарном приказу хромозома, тиме је изгубљено чак пола простора претраживања. У Матлаб програму постоји развијен графички интерфејс за имплементацију генетских алгоритама (Слика 4).



Слика 4. *ГА оптимтоол – софтвер Matlab*

3.2. КУКУ ПРЕТРАГА

Познати зов птица кукавица у рано пролеће је инспирисао многе поете и композиторе. Звук који је тако специфичан, део је љубавне игре између мужјака и женке у време када се мужјаци врате након зимске сеобе у топлије крајеве. Логичан завршетак ове игре је нова генерација птица која је настала паразитирањем кукавичјих јаја у гнездима других птица. Међународни истраживачки тим је баш у том паразитском понашању кукавица пронашао инспирацију за нову методу оптимизације. Xin-She Yang, са универзитета Кембриџ у Енглеској и Саш Деб са Високе техничке школе Раман, у Индији су 2009. године утврдили нови метод за решавање проблема оптимизације [11].

Кукавица свака два дана снесе по једно јаје, односно за време парења отприлике двадесет јаја. Већ пре него што јаје дозре, женка тражи одговарајуће гнездо. Када пронађе птицу која ће да буде „старатељ“ њеном потомству, посматра је још при градњи гнезда. Ако је то гнездо незгодно позиционирано, најпре снесе јаје на земљу, а затим га у кљуну преноси у гнездо. Ако је гнездо циљане птице пуно, кукавица зна да избаци неко јаје из гнезда или да га, чак, поједе, да би направила место за своје јаје. Она снесе јаје ноћу и брзо одлеће чим је посао обављен. Птице „старатељи“ нису благонаклоне према кукавицама, али ипак не могу да ускрате негу њеном јајету или младунчету.

Љуску јајета кукавица често копира у боју јајета птица-домаћина (мимикрија) да кукавичје јаје не би било избачено из гнезда. Осим тога и величина и облик су идентични онима код домаћина. Кукавица подмеће своја јаја припадницама 162 врста птица, од сиђушног краљића, преко црвендаћа, беле пастирице, језерског врапца или царића, до грлица и голубова [12].

Подметање не успева увек и не преживљавају сви птићи кукавице. Само око 62% подметнутих јаја примају невољни нови родитељи.

3.2.1. Алгоритам Јанга и Деба

Куку претрага (Cuckoo search – CS) се базира на следећим претпоставкама:

Свако јаје у гнезду представља решење, док кукавичје јаје представља оптимално решење. Циљ је да се добије ново и потенцијално боље решење (кукавичје јаје) да би се заменило не тако добро решење у гнезду. У најпростијој форми, свако гнездо има само једно јаје. Алгоритам може да се прошири на компликованије случајеве, где већи број јаја у гнезду представља скуп решења.

CS се базира на три идеализована правила:

- Свака кукавица полаже само једно јаје у тренутку времена које полаже у случајно одабрано гнездо.
- Најбоља гнезда са јајима високог квалитета ће опстати у наредној генерацији.
- Број расположивих гнезда домаћина је фиксан, а вероватноћа да ће кукавичје јаје бити препознато креће се у границама p_a є (0,1). Откривена јаја су искључена из следеће итерације.

Генерално, случајно променљива настаје када се одређеним случајним догађајима додељују бројеви или атрибути, што зависи од скале мерења, тако да је уз сваки додељени број или атрибут везана нека вероватноћа. Расподелу вероватноће неке случајно променљиве чине све вредности које та варијабла може имати, као и њене припадајуће вероватноће. Поступак претраживања решења (гнезда) који су усвојили Јанг и Деб се одиграва по поступку Левијеве расподеле вероватноће (Paul Pierre Lévy) [13]. Многе студије су показале да животиње и инсекти показују типичне карактеристике ове расподеле [14].

$$L(s)=\frac{1}{\pi}\int_0^{\infty}\cos(\tau s)e^{-\alpha\tau^{\beta}}d\tau,\ldots(0<\beta\le 2) \tag{10}$$

$$F_k\left[P_N(k)\right](x)=F_k\left\lfloor \exp\left(-N|k|^{\beta}\right)\right\rfloor(x) \tag{11}$$

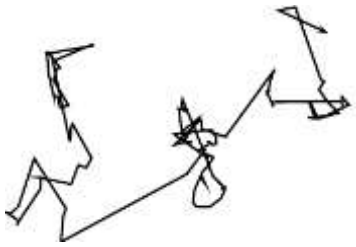
где је F_k – Фуријеова трансформација вероватноће P_N (k) за N-ту идентичну и независну случајну променљиву. За β=1 примењује се Кошијева расподела, док се за β=2 примењује Нормална расподела.

Упрошћена верзија Левијеве расподе може да се дефинише као:

$$L(s,\gamma,\mu)=\begin{cases}\sqrt{\frac{\gamma}{2\pi}}\exp\left\lfloor-\frac{\gamma}{2(s-\mu)}\right\rfloor\frac{1}{(s-\mu)^{3/2}}, & 0<\mu<s<\infty\\ 0, & \text{za sve druge slučajeve}\end{cases} \tag{12}$$

где је μ >0 минимални корак, а γ је фактор скалирања. Када s → β ,

$$L(s,\gamma,\mu)\approx\sqrt{\frac{\gamma}{2\pi}}\frac{1}{s^{3/2}} \tag{13}$$



Slika 5. *Левијев пут [11]*

Одабир одговарајућех корака итерације је битан за одрживост ефикасности тражења решења. Мали кораци дају мале промене, велики кораци доводе до великих скокова и чак и до удаљавања од простора са решењима. Левијев пут (Levy flights) је самим тим ефикаснији за проналажење решења него случајни пут. Варијанса Левијевог пута много брже расте него код линеарних веза или случајних путева. Са тачке примене, Левијев пут се састоји из два дела: избор случајног смера и стварање корака по Левијевој расподели (Слика 5).

Оригинално, алгоритам Јанга и Деба гласи (Слика 6):

begin
Objective function $f(\mathbf{x}),\ \mathbf{x} = (x_1, ..., x_d)^T$
Generate initial population of
n host nests $\mathbf{x}_i\ (i = 1, 2, ..., n)$
while *(t <MaxGeneration) or (stop criterion)*
Get a cuckoo randomly by Lévy flights
evaluate its quality/fitness F_i
Choose a nest among n (say, j) randomly
if $(F_i > F_j)$,
replace j by the new solution;
end
A fraction (p_a) *of worse nests*
are abandoned and new ones are built;
Keep the best solutions
(or nests with quality solutions);
Rank the solutions and find the current best
end while
Postprocess results and visualization
end

Слика 6. *Псеудокод CS [11]*

За стварање нових решења $x^{(t+1)}$ за кукавицу *i*, Левијев пут је:

$$x_i^{(t+1)} = x_i^t + \alpha \oplus Levy(\lambda) \tag{14}$$

где је α >0 величина корака. Обично је α =1. ⊕ означава улазну мултупликацију.

Левијев пут може бити и случајни пут када су случајни кораци добијени из Левијеве расподеле за велике кораке:

$$\text{Леви} \sim u = t^{-\lambda},\ (1<\lambda\leq 3) \tag{15}$$

са бесконачним варијансом и средњом вредношћу.

Овај алгоритам нашао је своје место у програмском пакету MATLAB, са случајним путем и случајном расподелом величине корака:

```
stepsize=rand*(nest(randperm(n),:)-nest(randperm(n),:))
new_nest=nest+stepsize.*K;
```

где је *K=rand(size(nest))>па* и *па* је степен налажења решења.

4. ПРИМЕР: ПРИМЕНА ГРУБИХ СКУПОВА ПРИ ИЗБОРУ ОПТИМАЛНЕ ЛОКАЦИЈЕ ЗА ФАБРИКУ БЕТОНА

Бетон као грађевински материјал, веома често, у већини радова које изводе грађевинска предузећа заузима главно место и по количини и по цени. Захтеви који се постављају у погледу квалитета, особина, количине и цене бетона намећу потребу за применом фабрика бетона – комплетних аутоматских постројења за производњу бетонске масе. Процес производње се завршава транспортом свежег бетона до места уградње, који треба, зависно од конкретних околности, да се решава на најекономичнији начин.

О свему овоме се водило рачуна када је од девет могућих варијантних решења, различитих карактеристика на подручју града требало изабрати нову локацију за будућу фабрику бетона једног грађевинског предузећа.

Подразумева се да фабрика бетона у свакој варијанти има исти материјални резултат у смислу трошкова производње, не подразумевајући ефекат услед положаја локације у односу на градилишта којима се транспортује бетон. Овај ефекат, преко цене бетона, утиче на целокупну грађевинску производњу у предузећу, а резултат је низа утицаја, који се, уз то, мењају у простору и времену (нпр. отварање новог градилишта, измена режима саобраћаја).

Усвојено је, на основу општег искуства, пет критеријума за вредновање варијантних решења:

- Удаљеност локације од градилишта и лакоћа приступа
- Величина локације
- Инфраструктура
- Цена куповине локације
- Цена уређења локације

Ови критеријуми представљају атрибуте услова, а решење применљива (или неприменљива) локација представља атрибут одлуке. Вредности атрибута изражене су лингвистички. Фабрика бетона изграђена на новој локацији треба да има оптималну повезаност са градилиштима, величину довољну за оптималну производњу бетона и повољну за организацију префабрикације, решену инфраструктуру и минимално ангажовање инвестиционих средстава за куповину и уређење локације.

У Табели 1 приказани су подаци о 9 разматраних локација. Табела представља *табелу атрибут – вредност* или *табелу одлучивања*. Колоне у табели обележене су атрибутима (особинама локације), а редови објектима (локацијама будуће фабрике бетона), док су записи у табели вредности атрибута. На тај начин сваки ред се може посматрати као информација о појединој локацији. На пример, локацију 7 у табели карактерише следећи скуп атрибута-вредности:

(Удаљеност локације, неповољна), (Величина локације, довољна), (Инфраструктура, неразвијена), (Цена локације, средња), (Цена уређења, висока) који представљају информацију о тој локацији.

Табела 1. *Табела одлучивања*

Локација	Атрибути услова					Атрибути одлуке
	Удаљеност локације	Величина локације	Инфраструктура	Цена локације	Цена уређења	Погодна локација
1	средња	средња	развијена	средња	ниска	да
2	средња	средња	развијена	средња	ниска	не
3	средња	недовољна	развијена	висока	средња	не
4	повољна	довољна	развијена	ниска	ниска	да
5	повољна	недовољна	неразвијена	ниска	висока	не
6	неповољна	недовољна	неразвијена	средња	висока	не
7	неповољна	довољна	неразвијена	средња	висока	не
8	неповољна	довољна	неразвијена	средња	средња	не
9	неповољна	средња	неразвијена	ниска	висока	не

Табела одлучивања генерише следеће скупове:

U = {локација 1, локација 2, локација 3, локација 4, локација 5, локација 6, локација 7, локација 8, локација 9}

A = {удаљеност локације, величина локације, инфраструктура, цена локације, цена уређења, погодна локација}

$V_{\text{удаљеност локације}}$ = {повољна, средња, неповољна}

$V_{\text{величина локације}}$ = {довољна, средња, недовољна}

$V_{\text{инфраструктура локације}}$ = {развијена, неразвијена}

$V_{\text{цена локације}}$= {висока, средња, ниска}

$V_{\text{цена уређења}}$ = {висока, средња, ниска}

$V_{\text{погодна локација}}$ = {да, не}

Из Табеле 1 се може издвојити осам елементарних скупова атрибута, односно A – скупова: један скуп објеката који се не разликују {1, 2} и преосталих седам скупова објеката који се разликују. То су доње и горње апроксимације класа одлуке Да {1, 4} и Не {2, 3, 5, 6, 7, 8, 9}.

Доња апроксимација класе одлуке Да: {4}

Горња апроксимација класе одлуке Да: {1, 2, 4}

Гранично подручје класе одлуке Да: BN_A(Да) = {1, 2}

Доња апроксимација класе одлуке Не: {3, 5, 6, 7, 8, 9}

Горња апроксимација класе одлуке Не: {1, 2, 3, 5, 6, 7, 8, 9}

Гранично подручје класе одлуке Не: BN_A(Не) ={1, 2}

Тачност апроксимације класа одлуке је α_A (Да) = 0.33 и α_A (Не) = 0.75

Гранично подручје одлука је, код обе класе одлука, састављено од два објекта 1 и 2. Ови објекти (локације) имају потпуно исте вредности атрибута услова (не разликују се), али

различите вредности атрибута одлуке. Из тог разлога, табела 1 није конзистентна, или, другим речима, одлука доносиоца одлуке је неконзистентна опису објеката атрибутима из А. Ово је чест случај у табелама одлучивања и може настати из неколико разлога. Пре свега, сасвим је могуће да нису све особине објеката "сачуване" у табели података. Такође, један од могућих разлога је и да доносилац одлуке или особа која врши класификацију атрибута одлуке промени свој став према атрибуту одлуке, у овом случају према појму "погодна локација", итд.

Следећи корак је састављање минималног подскупа независних атрибута, који гарантује исти квалитет класификација као цео скуп А, односно редуктора, као и језгра атрибута.

Проналажење редуктора и језгра атрибута ће бити урађено преко *матрице разликовања*. У циљу изналажења матрице разликовања, табелу одлучивања је често потребно нешто другачије груписати.

Табела 2 представља реалнији приказ табеле одлучивања, јер су неконзистентне класе одлука представљене као једна подељена класа (x_1 и x_1').

Табела 2. *Табела одлучивања*

Локација	Атрибути услова					Атрибути одлуке
	Удаљеност локације	Величина локације	Инфраструктура	Цена локације	Цена уређења	Погодна локација
x1	средња	средња	развијена	средња	ниска	да
x1'	средња	средња	развијена	средња	ниска	не
x2	средња	недовољна	развијена	висока	средња	не
x3	повољна	довољна	развијена	ниска	ниска	да
x4	повољна	недовољна	неразвијена	ниска	висока	не
x5	неповољна	недовољна	неразвијена	средња	висока	не
x6	неповољна	довољна	неразвијена	средња	висока	не
x7	неповољна	довољна	неразвијена	средња	средња	не
x8	неповољна	средња	неразвијена	ниска	висока	не

Матрица разликовања табеле података је симетрична матрица n × n са записима c_{ij}, і дата је у Табели 3.

Одговор на питање да ли се сви објекти разликују једни од других добија се применом конјункције свих записа у матрици разликовања.

Функција разликовања (Bool – ова функција) има следећу форму: f(UL, VL, I, CL, CU) = (VL ∪ CL ∪ CU) (UL ∪ VL ∪ CL) (UL ∪ VL ∪ I ∪ CL ∪ CU) (UL ∪ VL ∪ I ∪ CU) (UL ∪ VL ∪ I ∪ CU) (UL ∪ I ∪ CL ∪ CU) (UL ∪ VL ∪ CL ∪ CU) (UL ∪ I ∪ CL ∪ CU) (UL ∪ I ∪ CL ∪ CU) (UL ∪ VL ∪ I ∪ CL ∪ CU) (UL ∪ VL ∪ I ∪ CL) (UL ∪ VL ∪ I ∪ CL ∪ CU) (VL ∪ I ∪ CU) (UL ∪ VL ∪ I ∪ CL ∪ CU) (UL ∪ I ∪ CL ∪ CU) (UL ∪ I ∪ CL ∪ CU) (UL ∪ VL ∪ I ∪ CU) (UL ∪ CL) (UL ∪ VL ∪ CL) (UL ∪ VL ∪ CL ∪ CU) (UL ∪ VL) (VL) (VL ∪ CU) (VL ∪ CL) (CU) (VL ∪ CL) (VL ∪ CL ∪ CU)

(16)

Где су:

UL – Удаљеност Локације; VL – Величина Локације;I – Инфраструктура; CL – Цена Локације; CU – Цена Уређења

Табела 3. *Матрица разликовања*

	x1	x2	x3	x4	x5	x6	x7	x8
x1	-	VL, CL, CU	UL, VL, CL	UL, VL, I, CL, CU	UL, VL, I, CU	UL, VL, I, CU	UL, VL, I, CU	UL, I, CL, CU
x2	-	-	UL, VL, CL, CU	UL, I, CL, CU	UL, I, CL, CU	UL, VL, I, CL, CU	UL, VL, I, CL	UL, VL, I, CL, CU
x3	-	-	-	VL, I, CU	UL, VL, I, CL, CU	UL, I, CL, CU	UL, I, CL, CU	UL, VL, I, CU
x4	-	-	-	-	UL, CL	UL, VL, CL	UL, VL, CL, CU	UL, VL
x5	-	-	-	-	-	VL	VL, CU	VL, CL
x6	-	-	-	-	-	-	CU	VL, CL
x7	-	-	-	-	-	-	-	VL, CL, CU

Проблем проналажења подскупа атрибута који чува (обезбеђује) релацију неразликовања, применом матрице разликовања, своди се на проналажење импликаната једначине (16). Импликант је конјункција променљивих таква да, ако су те променљиве тачне, онда је и функција тачна. Посебно су значајни прости импликанти, који су, заправо, имликанти минималне величине.

Табела 4. *Табела одлучивања након редукције*

Локација	Атрибути услова		Атрибути одлуке
	Величина локације	Цена уређења	Погодна локација
1	средња	ниска	Да
2	средња	ниска	Не
3	недовољна	средња	Не
4	довољна	ниска	Да
5	недовољна	висока	Не
6	недовољна	висока	Не
7	довољна	висока	Не
8	довољна	средња	Не
9	средња	висока	Не

Једначина (16) се своди на :

$$f(UL, VL, I, CL, CU) = VL \cap CU \qquad (17)$$

Једначина (17) означава да атрибути величина локације и цена уређења представљају језгро атрибута и да се не могу елиминисати без губљења способности у апроксимацији одлучивања. Табела 4 представља табелу одлучивања након редукције.

Осим локација 1 и 2 чије су класе одлука неконзистентне, и локација 5 и 6, сви остали елементи се разликују.

Правила одлучивања генерисана из табеле 8 имају следећу форму:

IF (Величина локације, средња) & (Цена уређења, ниска) ⇒ THEN (Погодна локација, да)

IF (Величина локације, средња) & (Цена уређења, ниска) ⇒ THEN (Погодна локација, не)

IF (Величина локације, недовољна) & (Цена уређења, средња) ⇒ THEN (Погодна локација, не)

IF (Величина локације, довољна) & (Цена уређења, ниска) ⇒ THEN (Погодна локација, да)

IF (Величина локације, недовољна) & (Цена уређења, висока) ⇒ THEN (Погодна локација, не)

IF (Величина локације, довољна) & (Цена уређења, висока) ⇒ THEN (Погодна локација, не)

IF (Величина локације, довољна) & (Цена уређења, средња) ⇒ THEN (Погодна локација, не)

IF (Величина локације, средња) & (Цена уређења, висока) ⇒ THEN (Погодна локација, не)

5. БИБЛИОГРАФИЈА

[1] S. Rao, Engineering optimization, Theory and Practice, John Wiley & Sons, New York, 1996, 894.

[2] Z. Pawlak, Rough Sets, International Journal of Computer and Information Science, 11, 341–356,1982.

[3] G. Ćirović, G., D. Plamenac, Rough sets – application in the construction industry, Operational Research Society, Belgrade, 2005, 224.

[4] S. N. Sivanandam, N. Deepa, Introduction to genetic algorithms, Springer-Verlag, Berlin, 2008, 462.

[5] K. Miettinen, P. Neittaanmaki, M. M. Makela, P. Periaux, Evolutionary algorithms in engineering and compurter science, John Wiley & sons, New York, 1999, str. 483.

[6] C. R. Reeves, J. E. Rowe, Genetic Algorithms: Principles and Perspectives A Guide to GA Theory, Kluwer Academic Publishers New York, 2002, 319.

[7] J. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan, 1975.

[8] R. Riolo, U. M. O'Reilly, T. McConaghy, Genetic programming, Theory and Practice, Springer Science+Business Media, 2010.

[9] M. Srinivas. L. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on System, Man and Cybernetics, 24(4), pp.656–667, 1994.

[10] X. S. Yang, S. Deb, Cuckoo search via Lévy flights, World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. str. 210–214., 2009.

[11] A. E. Brehm, Живот животиња, Моно Мањана, 2004.

[12] M. Shlesinger, G. M. Zaslavsky, U. Frisch, Lévy Flights and Related Topics in Physics, New York: Springer-Verlag, 1995.

[13] C. Brown, L. S. Liebovitch, R. Glendon, Levy flights in Dobe Juhoansi foraging patterns, Human Ecol., 35, 129–138, 2007.