УНИВЕРЗИТЕТ У БЕОГРАДУ

МАТЕМАТИЧКИ ФАКУЛТЕТ

Душан Милосављевић

Метода променљивих околина за решавање проблема континуалне оптимизације

мастер рад

Београд , 2018

Подаци о ментору и члановима комисије

Ментор:

др Милан Дражић,
ванредни професор,
Универзитет у Београду, Математички факултет

Чланови комисије:

др Милан Дражић,
ванредни професор,
Универзитет у Београду, Математички факултет

др Александар Савић,
ванредни професор,
Универзитет у Београду, Математички факултет

др Зорица Станимировић,
ванредни професор,
Универзитет у Београду, Математички факултет

Датум одбране:

Метода променљивих околина за решавање проблема континуалне оптимизације

Резиме

Класичне методе локалне претраге у пракси често не доводе до најбољег, глобалног решења неког оптимизационог проблема. За разлику од њих, метода променљивих околина (енгл. Variable Neighborhood Search, VNS) је метахеуристички приступ заснован на систематичној промени околина унутар простора решења у току претраге, у циљу проналажења бољег решења. Метода је први пут изложена у литератури 1997. године и првобитно се користила за решавање проблема комбинаторне оптимизације, а данас се широко примењује и на проблеме континуалне оптимизације. Ефикасност методе у великој мери зависи од величине и геометрије коришћених околина, од начина случајног избора тачака у фази размрдавања VNS методе као и начина претраживања околина у фази локалног претраживања. Претраживањем погодно изабраних околина, VNS је ефикаснија од многих других метахеуристичких метода при решавању бројних проблема глобалне оптимизације.

У раду ће бити приказан алгоритам методе променљивих околина, различите типове околина које се користе, као и методе локалног претраживања које су се показале ефикасним у пракси. Затим ће бити представљена имплементација алгоритма развијена у овом раду, резултати за проблеме различитих димензија који су познати из литературе, као и однос тих резултата са познатим резултатима других метахеуристичких метода.

Кључне речи: континуална оптимизација, глобална оптимизација, метода променљивих околина, метахеуристике.

Научна област: Математика

Ужа научна област: Оптимизација

Садржај

1 Увод

1.1 Оптимизациони проблем

Уопштено речено, оптимизација је процес проналажења решења које је *најбоље* (у зависности од контекста проблема) при одређеним условима. Задатак математичке оптимизације се може формулисати на следећи начин [18]:

$$\min_{x \in X} f(x), \tag{1.1}$$

где је $X \subseteq S$. Скуп S називамо простором решења, док је X подскуп простора решења, чији елементи задовољавају извесна ограничења, дефинисана у задатку. Реалну, ограничену функцију $f : S \to \mathbb{R}$ називамо функција циља. У случају континуалне (непрекидне) оптимизације, $S = \mathbb{R}^n$. Уколико је S коначан или пребројив скуп, тада говоримо о комбинаторној (дискретној) оптимизацији.

Уколико пронађемо $x^* \in X$ такво да је $f(x^*) \leq f(x)$, за свако $x \in X$, тада кажемо да је x^* тачка глобалног минимума функције f, односно оптимално решење проблема (1.1), а вредност $f(x^*)$ је глобални минимум, тј. вредност функције циља која одговара оптималном решењу. Ако постоји $\epsilon > 0$ такво да за неку тачку x^* важи да је $f(x^*) \leq f(x)$, за свако $x \in X$ и $||x^* - x|| \leq \epsilon$, где је $|| \cdot ||$ функција норме у простору $\mathbb{R}^n$, тада је x^* тачка локалног минимума, а $f(x^*)$ локални минимум. Уколико постоји само један локални минимум, он је уједно и глобални минимум. Међутим, у пракси често постоји много локалних минимума неког проблема и задатак је пронаћи најмањи међу њима. Тачака глобалног минимума, односно решења које може имати исту (минималну) вредност функције циља такође може бити више, али је обично задатак пронаћи само једну од њих.

С обзиром на то да је $\max_{x \in X} f(x) = -\min_{x \in X}(-f(x))$, проблем максимизације се може свести на проблем минимизације, па је довољно разматрати само проблем тражења минимума функције f.

Потреба за решавањем проблема оптимизације се јавља у многим областима: у инжењерству, обрадама података, финансијском планирању, процени ризика, научном моделирању, транспорту робе, хемији итд. Главни проблем је у томе што најчешће постоји много (често и бесконачно много) локалних минимума, посебно у случају проблема великих димензија (са великим бројем променљивих), а такви проблеми су чести

у пракси. Чак и у случају проблема мањих димензија, класичне (егзактне) методе које гарантују налажење најбољег решења често захтевају сувише рачунарског времена или меморије да би се решење нашло (под условом да оно постоји). Многи проблеми и дискретне и континуалне оптимизације спадају у класу проблема за које није познат алгоритам полиномијалне сложености за њихово решавање (NP-комплетни проблеми) [43, 63]. Чак и за случај дискретне оптимизације, где је скуп X често коначан, није пожељно применити егзактне методе јер са повећањем броја променљивих се број елемената скупа драстично повећава па идеја тражења глобалног минимума неком егзактном методом постаје непрактична. Такође, некад није могуће применити егзактне методе и због тога што математички модел којим је дефинисан сам проблем има превише ограничења. Због тога, развијене су другачије, метахеуристичке методе које не гарантују да ће се глобални минимум наћи, али да се у релативно кратком времену извршавања на рачунару може наћи решење које је у извесном смислу "довољно добро".

1.2 Преглед најпознатијих метахеуристика

Хеуристике су, у најопштијем смислу, робустне методе и технике итеративне природе засноване на искуству које брзо дају решење које је, за потребе корисника, довољно блиско оптималном решењу. Метахеуристика је скуп алгоритамских концепата који се користе за дефинисање хеуристичких метода које су примењиве на широк спектар проблема. Као врсте апроксимативних метода оптимизације, специјалне хеуристике (енгл. specific heuristics) се уско прилагођавају конкретном проблему, уколико би се применило на неки други проблем, морало би се све испочетка имплементирати. За разлику од њих, метахеуристике су општији концепти који се могу применити на разне класе проблема оптимизације. Може се посматрати као генерализована методологија која може бити искоришћена као стратегија у конструисању фундаменталне хеуристике за решавање оптимизационих проблема. [63].

Неке метахеуристичке методе, попут методе променљивих околина, у ужем или ширем смислу су засноване на принципу **локалног претраживања** (енгл. Local Search, LS) [11]. Овај принцип подразумева конструисање неке структуре назване *околине* (или више њих) у простору реше-

ња. Наиме, сваком елементу $x \in S$ се додељује неки подскуп $N(x) \subseteq S$, који се назива *околина* од x, а њени чланови су *суседи* од x. Претраживање полази од произвољне тачке (решења) из простора S, која се прогласи за најбоље решење. Затим се у свакој итерацији претражује околина најбољег решења и у њој налази, према неком дефинисаном правилу, сусед који представља следеће решење. Уколико се такав сусед не може наћи, или ако је испуњен неки други критеријум заустављања, метода стаје и последње генерисано текуће решење се узима за апроксимацију оптималног решења.

Главни недостатак овог принципа приликом решавања проблема глобалне оптимизације (1.1) је тај што се у процесу претраге често не могу избећи замке локалних минимума, односно „заглављивања" у околини истог. Зато се, између осталог, и одређује посебна структура околине или више њих, као и критеријум за избор следећег суседа у конкретној околини, а за неке метахерустике се уводе неке друге идеје за превазилажење овог проблема.

У даљем тексту су наведене неке од најпознатијих метахеуристика, са њиховим основним идејама, док се детаљнији описи метахеуристика и њихових модификација могу наћи у [5, 11, 24, 44, 58, 59].

1.2.1 Симулирано каљење

Симулирано каљење (енгл. Simulated Annealing, SA) [11] је метахеуристика која је спој детерминистичке и пробабилистичке стратегије претраживања простора решења. На почетку се генерише неко почетно решење x, оно постаје текуће решење, а затим у свакој итерацији ова метода бира на случајан начин неког суседа из околине тренутног решења, прихватајући га као ново, текуће решење, не само у случају побољшања, већ и у случају погоршања вредности функције циља, али са извесном вероватноћом. Та вероватноћа се постепено мења током итерација, по унапред дефинисаном закону. На тај начин се замка локалних минимума може избећи у већем броју случајева и тако повећати шанса за добијање квалитетног решења.

Основна идеја симулираног каљења заснива се на аналогији са законима и алгоритмима статичке термодинамике. Први алгоритам који симулира

процесе термодинамике је изложен у раду [49]. Он симулира процес контролисаног хлађења (каљења) неког растопљеног материјала до постизања кристализованог чврстог стања. Овај алгоритам на случајан начин мења тренутну конфигурацију атома материјала, што доводи до промене његове унутрашње енергије. Ако се енергија смањује, нова конфигурација се прихвата, а ако се повећава, нова конфигурација се прихвата са извесном вероватноћом која се одређује према Болцмановом термодинамичком закону. Полазећи од неке случајно генерисане конфигурације атома, овај процес се понавља задати број пута на свакој температури, при чему се температура постепено смањује (тзв. *шема хлађења*). Уколико је ово смањивање довољно споро, закони термодинамике кажу да ће материјал тежити да постигне стање минималне енергије, тј. да очврсне у правилну кристалну структуру.
Деценије су прошле да би у [21] и [42] било показано да ови основни принципи могу бити примењени и на проблеме оптимизације, при чему допустиво решење одговара једној конфигурацији атома материјала, вредност функције циља његовој унутрашњој енергији, а потенцијални прелаз у суседно решење могућој промени стања енергије. Од тада се ова методологија и примењује за решавање проблема оптимизације. СК је засновано на локалном претраживању. Оно увек прихвата боље решење, док лошије прихвата са извесном вероватноћом која зависи од тога колико је ново решење лошије од текућег и која опада са опадањем температуре. Температура има задатак да контролише флексибилност прихватања погоршања функције циља, а тиме и претраживање простора X. Критеријум заустављања, као и начин дефинисања шеме хлађења, су особености самог алгоритма симулираног каљења и од њиховог дефинисања, као и од већ поменутих особина локалног претраживања, зависи и ефикасност самог алгоритма. Требало би напоменути да се не може за сваки проблем дефинисати општа правила за избор поменутих параметара, тако да се до адекватних вредности параметара за решавање конкретног проблема или групе сличних проблема симулираног каљења долази обимним експериментисањем.

1.2.2 Табу претраживање

Основни концепт табу претраживања (енгл. Tabu Search,TS) за решавање проблема оптимизације предложио је Гловер, 1985. у раду [25], док

су сличне идеје независно развили Хансен и Јаумард, 1987. године [29]. Табу претраживање се базира на идеји локалне претраге и користи тзв. *адаптивну меморију*, тј. памћење одређених информација о претходним фазама процеса претраживања, које затим утичу на избор следећих тачака у итеративном процесу [11]. Наиме, у свакој итерацији $n = 1, 2, ...$ чува се историја H претходног претраживања, тј. запис који памти унапред изабране карактеристике неких од претходно генерисаних тачака. Околина $N(x_n)$ тренутне тачке x_n се модификује у зависности од историје H и на тај начин добијемо околину $N(x_n, H)$ који представља скуп суседа за следећу тачку претраживања x_{n+1}. Пошто овај скуп може бити понекад сувише велики (посебно при решавању проблема континуалне оптимизације), узима се његов „мали" подскуп $N^{'}(x_n)$. Наредна тачка итеративног низа x_{n+1} се одређује као *најбоља* у простору $N^{'}(x_n)$ у односу на неку функцију процене $f(x, H)$ која, општем случају, може бити измењена и која зависи од историје H. Погодним дефинисањем $N(x_n, H)$ и $f(x, H)$ могу се, као кандидати за следећу тачку претраживања, фаворизовати они суседи тачке x_n који имају или немају карактеристике садржане у историји H. Табу претраживање дозвољава и успињуће помаке, тј. када долази до погоршања функције циља и тако избегава заглављивање у локалним минимумима.

Најважнији део ове методологије представља дефинисање одговарајуће адаптивне меморије, тј. начин формирања и ажурирања историје H. У основној верзији табу претраживања, постоји само један тип меморије - *краткорочна меморија*, код које у свакој итерацији историја H памти карактеристике тачака генерисаних у одређеном броју претходних итерација (тзв. табу листа). Тако дефинисана меморија има улогу да, забрањујући неке суседе из околине тренутне тачке, спречи враћање претраге током одређеног броја наредних итерација на неке од раније генерисаних тачака. На тај начин се омогућава ширење процеса претраге у неке нове области допустивог простора X. Често краткорочна меморија није довољна да се постигну две фазе које се периодично смењују - локална интензификација (задржавање у некој „доброј" области простора X) и глобална диверзификација (претраживање у неке нове, неистражене делове простора X). Зато се у развијенијим верзијама табу претраживања, оваква меморија комбинује са неким облицима дугорочнијег меморисања, у циљу побољшања и убрзања претраге [11]. Детаљнији приказ главних карактеристика дугорочне меморије може се наћи у [26] и [16].

1.2.3 Генетски алгоритми

Генецки алгоритми (енгл. Genetic algorithms, GA) су метахеуристика које симулирају процес генетске еволуције једне популације јединки под дејством окружења и генецких оператора. Сваку јединку у популацији карактерише хромозом, који се мозе сматрати њеним јединственим кодом. Оне јединке из популације које су више прилагођене окружењу, међусобно се и даље репродукују (применом оператора на њихове хромозоме) и тако ствара нова генерација јединки, прилагођенија од претходне. Овај процес се наставља, при чему се из генерације у генерацију повећава просечна прилагођеност чланова популације.
Генетски алгоритми се могу користити за решавање разних проблема комбинаторне и континуалне оптимизације [40]. ГА је заснован на општим принципима генетике, примењене математичким језиком: сваком решењу из скупа X разматраног проблема додељује се, на тачно дефинисан начин, код (хромозом) решења. Код може бити коначан низ симбола над неком азбуком, матрица, дрвоидна структура или нека комбинација (нпр. низа и матрице). Начин кодирања зависи од карактеристика разматраног проблема оптимизације. Скуп кодова свих решења из X чини *простор кодираних решења* $\overline{X}$. ГА итеративно врши претраживање у простору кодираних решења $\overline{X}$. У свакој итерацији алгоритам генерише одређен скуп тачака из $\overline{X}$ који представља *популацију*. За сваку тачку x_i у n-тој итерацији се одређује њена „прилагођеност" $F(x_i)$, где је F такозвана *функција прилагођености*. Она може бити баш једнака функцији f. Ако је f „скупа" за рачунање, онда се за F узима нека функција која је једноставна и „јефтина" за рачунање у временском смислу, али тако да добро одсликава квалитет решења. Затим се на случајан начин бирају оне тачке из $\overline{X}$ које ће учествовати у стварању нове популације. То се реализује оператором селекције који фаворизује боље прилагођене јединке, тј. јединке са већом вредношћу функције F, па се на такве јединке делује операторима, дефинисаним по узору на генетске трансформације у биолошким системима. У основној верзији алгоритма примењују се два таква оператора - *укрштање* и *мутација*. Оператор укрштања је поступак у коме се случајно узајамно размењују делови кодова (хромозома) два решења (родитеља) из $\overline{X}$ и тако доби-

јају кодови два нова решења (деце). Овакав оператор се примењује на сваки пар родитеља (који су изабрани оператором селекције), али на случајан начин са унапред задатом вероватноћом укрштања p_u. Оператор мутације врши промену садржаја кода неког решења из $\overline{X}$ случајном заменом појединих симбола овог кода са неким другим симболима из азбуке симбола. Овакав оператор се примењује на свако дете са унапред задатом вероватноћом мутације p_m. Мутација игра секундарну улогу у формирању јединки нове генерације, па би вероватноћа p_m требало да буде веома мала (обично реда 10^{-3}). Укрштањем добро прилагођених јединки добијају се јединке које су прилагођене бар колико и родитељи, ако не и боље, док се мутацијом избегава доминација јединки са истим особинама. На тај начин се генерише нова популација која представља следећу генерацију у процесу претраживања. Код основне верзије ГА претраживање обично стаје када се достигне неки унапред задат максималан број итерација. Детаљније карактеристике овог алгоритма могу се наћи у [10, 14, 27].

1.2.4 Оптимизација мрављим колонијама

Идеју имитирања понашања и кретања мрава у колонијама за дизајнирање методе која се може користити за решавање проблема у оптимизацији је први пут предложио Дориго у [15] (енгл. Ant colony optimization, ACO). ACO је заснована на принципима кретања мрава у потрази за храном и како налазе пут за повратак у скровиште. Приликом кретања мрави испуштају и остављају траг, тзв. "феромоне". Циљ остављања трагова је навођење осталих мрава на „позељан" пут где би требало да се крећу у потрази за храном. Један мрав пут бира на основу јачине трага феромона на њему. Процес овог алгоритма примењен на оптимизацију је следећи: први корак је иницијализација, која се састоји у одређивању броја мрава m (вектора у скупу допустивих решења x_k, $k = 1, 2, ..., m$) који ће бити укључени у претрагу, њихов иницијални распоред (обично се распоређују на случајан начин унутар простора решења), почетна јачина трага феромона за сваког мрава, као и најближег мрава до скровишта (почетно најбоље решење x_{best}). Затим се итеративно $n = 1, 2, ...$ врши модификација и ажурирање за сваког мрава (сваког вектора x_k) која зависи од трага феромона, који се за сваког

мрава динамички мења током претраге. Одређује се текуће најбоље решење након измена позиција мрава (измена вектора претраге x_k), а затим се за сваког мрава ажурира његова јачина трага феромона, прво тзв. *фазом евапорације* (енгл. evaporation phase), односно редукцијом јачине трага по угледу на биолошке трансформације, а онда и ажурирањем трага на начин који зависи од генерисаних решења x_k, $k = 1, ..., m$ (енгл. reinforcement phase). Један од предложених начина [63] је да се траг феромона ажурира само за p мрава који су најближи скровишту (p најбољих текућих решења међу векторима x_k), додавањем неког позитивног броја Δ на текућу јачину. На тај начин се даје информација о жељи даљег кретања мрава (решења дуж допустивог простора решења). Критеријум заустављања је обично максималан дозвољен број итерација.

1.2.5 Оптимизација колонијом пчела

Оптимизација колонијом пчела (енгл. Bee Colony Optimization, BCO), која као и ACO спада у групу метахеуристичких метода које су засноване на тзв. интелигенцији роја (енгл. swarm intelligence), је инспирисана понашањем пчела у потрази за храном. Предложили су је Лучић и Теодоровић 2001. године [46]. Пчеле у природи трагају за храном и доносе узорке нектара у кошницу где се испитује његов квалитет. Квалитетнији нектар је праћен плесом пчела којим се указује на правац и даљину одакле је донет. Свака пчела се са неком вероватноћом одлучује или да прати траг где је већ пронађен добар нектар, да игром „рекламира" место хране и тиме „регрутује" друге пчеле да прате тај траг или да наставља да трага за новим извором хране. Метода се прво састоји од иницијализације, односно свакој пчели (B - број пчела као улазни параметар) се додељује једно почетно (празно) решење. Свака итерација методе се састоји из две главне фазе: предња фаза или лет унапред (енгл. forward pass) и задња фаза или лет уназад (енгл. backward pass). У предњој фази, пчеле претражују цео простор допустивих решења, примењујући унапред дефинисан број помераја, чиме се конструишу (или побољшавају) позиције пчела (текућа решења, тј. тачке које се користе у претраживању). У задњој фази, све пчеле садрже информације о квалитету решења осталих пчела. Идеја плеса пчела је у алгоритму имплементиран рачунањем функције циља. Када се израчунају све вредности функције циља у текућим решењима за сваку пчелу,

свака пчела одређује, са извесном вероватноћом, да ли остаје *лојална* свом решењу или не. Пчеле са бољом вредношћу функције циља имају већу шансу да задрже текуће решење. Пчелама, које остану лојалне свом решењу, тзв. *регрутима* (енгл. recruiters), та решења разматрају преостале пчеле. Оне пчеле које не остану лојалне постају *неопредељене* (енгл. uncommited) и морају да преузму неко од решења лојалних пчела. Одлука које ће се решење преузети се доноси уз извесну вероватноћу, при чему боља решења имају већу шансу да буду решења нелојалних пчела. Тако у свакој итерацији долази до поделе пчела на лојалне (recruiters) и нелојалне (uncommited). Разлика између ACO и BCO је у томе што комуникација између мрава је индиректна и заснована на појединачним компонентама решења, а пчеле комуницирају директним поређењем (парцијалних) решења. Детаљнији опис алгоритама заснованих на овој идеји се може наћи у [13, 47].

2 Метода променљивих околина

2.1 Основне идеје и варијанте методе променљивих околина

Методу променљивих околина (енгл. Variable Neighborhood Search, VNS) су 1997. године предложили Младеновић и Хансен у раду [53], а идеја је први пут изложена 1995.године [50]. Метода је првобитно развијена за проблеме дискретне оптимизације, а данас налази широку примену не само за проблеме дискретне, већ и за проблеме континуалне оптимизације. Основна идеја ове методе заснована је на систематичној промени околина, настојећи истовремено да се нађе локални минимум, као и могућности да се из њега „побегне", уколико то није глобални минимум проблема. VNS се заснива на три запажања [18, 34]:

1) Локални минимум у једној околини није нужно локални минимум за неку другу околину.

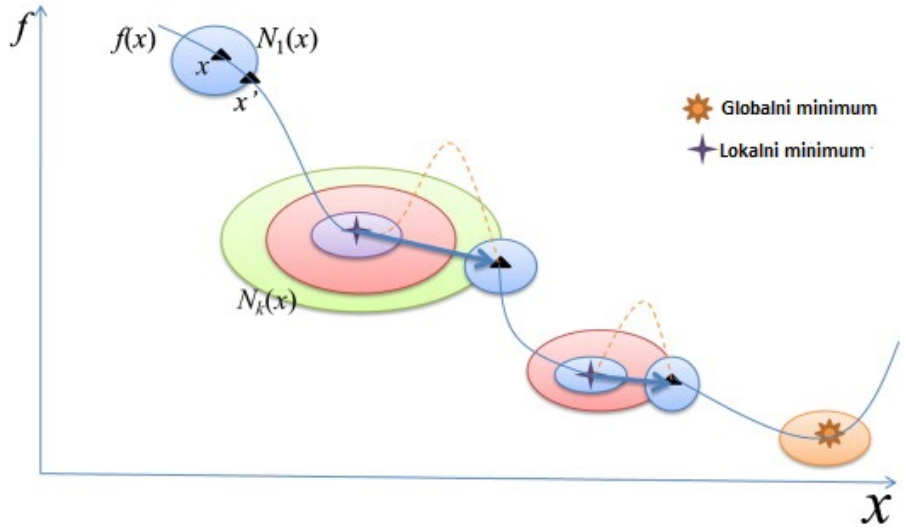
2) Глобални минимум је локални минимум у свим околинама.

3) За велики број проблема локални минимуми су релативно близу једни другима.

Последња ставка, која је чисто емпиријска, указује на то да локални минимум често садржи информацију о глобалном и због тога се јавља потреба за истраживањем околина локалних минимума. Ова три запажања се могу користити детерминистички, стохастички или комбиновано, чиме се добијају различите модификације VNS-а. За било коју варијанту VNS-а, потребно је у простору X из (1.1) дефинисати коначан скуп околина које означавамо са N_k, $k = 1, 2, ..., k_{max}$, где $N_k(x)$ означава k-ту околину тачке x. Најучесталији случај је $N_1 \subset N_2 \subset N_3 \subset \cdots \subset N_{k_{max}}$. На Слици 1.1 илустрован је начин претраживања околина код методе променљивих околина (преузето из [1]).
У даљем тексту наведене су неке од основних варијанти VNS методе, док се више детаља може наћи у [30, 31, 34, 35, 53].

2.1.1 Метода променљивог спуста

Како локални минимум у једној околини није обавезно локални минимум у некој другој околини, промена околине се може одвијати при-



Слика 1: Илустрација примене методе променљивих околина

ликом локалне претраге, у потрази за глобалним минимумом. Варијанта методе променљивих околина која се заснива на овој идеји назива се *метода променљивог спуста* (енгл. Variable Neighborhood Descent, VND) [18, 31, 34]. Након што се одабере систем околина као и њихов редослед бирања $N_1, N_2, ..., N_{k_{\max}}$, метода креће извршавање од почетне тачке x која се поставља као најбоље решење $\overline{x}$ у првој итерацији, а затим врши локално претраживање, неком методом локалне претраге, у првој околини $N_1(\overline{x})$. Нека је пронађени локални оптимум у околини $N_1(\overline{x})$ означен са ш'. Уколико вредност функције циља у тачки $x^{'}$ није боља од $\overline{x}$, индекс k се увећава за 1 и претрага се наставља у наредној околини $N_k(\overline{x})$, све док је $k \leq k_{\max}$. Сваки пут када је пронађен локални минимум x' у околини $N_k(\overline{x})$ бољи од $\overline{x}$, тј. такво да је $f(x^{'}) < f(\overline{x})$, оно се проглашава за најбоље решење ($\overline{x} \leftarrow x^{'}$), k се поставља на 1 и претрага се наставља у околини $N_1(\overline{x})$. У свакој итерацији, кад год је решење локалне претраге $x^{'}$ у околини $N_k(\overline{x})$ лошије од тренутно најбољег $\overline{x}$, индекс k се увећава за 1 и претрага се наставља у околини $N_k(\overline{x})$. Процес се зауставља када обиђемо све околине тренутног решења $\overline{x}$. VND метода као резултат враћа вредност функције циља у тачки $\overline{x}$ и ту вредност проглашавамо за решење нашег проблема.

VND је детерминистичка метода, што значи да у свом изворном облику не садржи идеје случајног избора. Приликом локалног претраживања околина VND најчешће користи стратегију *најбољег побољшања* (енгл. Best improvement strategy), односно тачка $x^{'}$ која се добије приликом локалне претраге у некој околини је и најбоље побољшање. Друга могућ-

ност је примена стратегије *првог побољшања* (енгл. First improvement strategy) која у конкретној околини проналази прво решење $x^{'}$ које је боље од тренутног и претрагу даље наставља посматрајући околине $N_k,\ k = 1, 2, ...k_{max}$ тачке $x^{'}$. Стратегија првог побољшања се користи у случајевима када је стратегија најбољег побољшања сувише временски захтевна, нпр. код сложених проблема где је тешко наћи допустиве суседе или околина садржи сувише велики број допустивих решења које би требало све обићи.
Псеудо-шематски приказ методе променљивог спуста представљен је Алгоритмом 1.

Алгоритам 1. Основна структура методе VND.

/ Иницијализација */*
Дефинисати низ околина $N_k,\ k = 1, 2, ..., k_{\max}$;
На случајан начин изабрати почетно решење $x \in X$ и $\overline{x} \leftarrow x$;
понављај наредне кораке **док** није испуњен критеријум заустављања
1: Поставити $k \leftarrow 1$;
понављај наредне кораке **док** $k > k_{\max}$
/ Истраживање околине */*
Пронаћи најбоље решење x' у околини $N_k(\overline{x})$;
/ Да ли се померити? */*
Ако $f(x') < f(\overline{x})$ **онда**
Поставити $\overline{x} \leftarrow x'$, $f(\overline{x}) \leftarrow f(x')$ и **иди на** 1;
иначе
Поставити $k \leftarrow k + 1$;
крај
крај
крај
Стоп. Тачка $\overline{x}$ је апроксимација решења проблема.

2.1.2 Редукована метода променљивих околина

Приликом употребе методе променљивих околина, обично највише времена захтева тражење локалног минимума у конкретној околини, као и „бег" из замке локалног минимума једном када је пронађен. Због тога је у [31, 34] предложена нова варијанта VNS-а, тзв. *редукована метода променљивих околина* (енгл. Reduced Variable Neighborhood Search, RVNS), која

у потпуности одбацује тражење најбољег решења у конкретној околини, већ тражи само прво побољшање функције циља $f(x')$ у некој тачки x', да би након тог побољшања наставила даље претрагу у околини тачке x'. RVNS метода се најчешће користи за добијање квалитетних решења која даље представљају почетна решења за неке друге метахеуристичке или егзактне методе. Међутим, поставља се питање како се кретати кроз простор решења, односно дефинисати како у конкретној околини изабрати решење које ће бити упоређено са текућим најбољим решењем, у погледу вредности функције циља. Најједноставнији приступ је померати се на случајан начин. Наиме, након дефинисања структуре околина N_k, $k = 1, 2, ..., k_{max}$ и одређивања почетног решења x (постављајући га у почетном тренутку као тренутно најбоље решење, тј. $\overline{x} \leftarrow x$), из прве околине $N_1(\overline{x})$ се на случајан начин бира тачка x'. Затим се пореде вредности функције циља у тачки x' и у тачки $\overline{x}$. Уколико је $f(x') < f(\overline{x})$, постављамо за ново тренутно најбоље решење x' ($\overline{x} \leftarrow x'$), k се враћа на 1 и претрага се наставља почевши од прве околине $N_1(\overline{x})$. Уколико је $f(x') \geq f(\overline{x})$, индекс k се увећава за 1 и претрага се наставља у наредној околини тачке $\overline{x}$. Поступак случајног избора тачке x' из околине $N_k(\overline{x})$ назива се *размрдавање* (енгл. shaking). Када се претраже све околине, тј. када k достигне вредност k_{max}, RVNS метода наставља процедуру претраге, постављајући $k = 1$ и претражујући околине изнова док се не постигне побољшање или неки критеријум заустављања (нпр. максимално време извршавања, максималан број итерација, максималан број итерација између два побољшања итд.). Псеудо-шематски приказ методе RVNS дат је Алгоритмом 2.

Алгоритам 2: Основна структура методе RVNS.

/* *Иницијализација* */
Дефинисати низ околина N_k, $k = 1, 2, ..., k_{\max}$;
Дефинисати критеријум заустављања;
На случајан начин изабрати почетно решење $x \in X$ и $\overline{x} \leftarrow x$;
понављај наредне кораке **док** није испуњен критеријум заустављања
1: Поставити $k \leftarrow 1$;
понављај наредне кораке **док** није $k > k_{\max}$
/* *Размрдавање* */
На случајан начин изабрати тачку x' у околини $N_k(\overline{x})$;
/* *Да ли се померити?* */
Ако $f(x') < f(\overline{x})$ **онда**

Поставити $\overline{x} \leftarrow x'$, $f(\overline{x}) \leftarrow f(x')$ и **иди на** 1;

иначе

Поставити $k \leftarrow k + 1$;

крај

крај

крај

Стоп. Тачка $\overline{x}$ је апроксимација решења проблема.

2.1.3 Основна варијанта методе променљивих околина

Код VND методе је примарно било тражење локалног минимума, док је код RVNS варијанте најважнија идеја случајног избора тачке, односно *размрдавање*, која нам омогућава да претражимо и најудаљеније области подскупа X скупа решења, под условом да дефинисан скуп околина то дозвољава. Основна варијанта методе променљивих околина (енгл. Basic Variable Neighborhood Search, BVNS) користи идеје претходне две методе. Приказ псеудокода основне варијанте VNS методе дат је Алгоритмом 3. Након што се дефинишу околине $N_1, N_2, ..., N_{k_{max}}$, почетна тачка x и одреди критеријум заустављања, приступа се процесу размрдавања, односно бира се случајна тачка x' из прве околине $N_1(\overline{x})$, а затим се примењује локална претрага у некој околини тачке x', која не мора нужно да буде иста околина која се користи у процесу размрдавања (чак не мора бити ни исте структуре). Поступком локалне претраге одређује се неки локални минимум x''. Уколико је $x = x''$ (што би значило да је претрага стигла „на дно долине"), примењује се локална претрага на следећу околину тачке x, тј. $N_2(x)$. Ако је $x \neq x''$ и $f(x'') \geq f(x)$, није пронађено боље решење од тренутног и претрага се наставља такође у околини $N_2(x)$. У случају да је $f(x'') < f(x)$, боље решење x'' се прихвата као тренутно најбоље и описани поступак се даље примењује око тачке x'', односно поставља се $x \leftarrow x''$ и $k = 1$. Када се обиђу све околине неке тачке, односно када се достигне k_{max}, претрага се наставља око исте тачке постављајући $k = 1$. Итеративно понављамо процедуру размрдавања и локалне претраге док се не испуни неки критеријум заустављања (нпр. максималан број итерација).

Алгоритам 3: Псеудо-шема методе BVNS.

/* *Иницијализација* */
Дефинисати низ околина N_k, $k = 1, 2, ..., k_{\max}$;
Дефинисати критеријум заустављања;
На случајан начин изабрати почетно решење $x \in X$ и $\overline{x} \leftarrow x$;
понављај наредне кораке **док** није испуњен критеријум заустављања
1: Поставити $k \leftarrow 1$;
понављај наредне кораке **док** није $k > k_{\max}$
/* *Размрдавање* */
На случајан начин изабрати тачку x' у околини $N_k(\overline{x})$;
/* *Локална претрага* */
Применити неку методу локалног претраживања у околини решења x' како би се добио локални минимум x'' проблема;
/* *Да ли се померити?* */
Ако $f(x'') < f(\overline{x})$ **онда**
Поставити $\overline{x} \leftarrow x''$, $f(\overline{x}) \leftarrow f(x'')$ и **иди на** 1;
иначе
Поставити $k \leftarrow k + 1$;
крај
крај
крај
Стоп. Тачка $\overline{x}$ је апроксимација решења проблема.

2.1.4 Општа варијанта методе променљивих околина

Ако се процедура локалне претраге у BVNS варијанти замени VND методом, добија се Општа метода променљивих околина (енгл. General Variable Neighborhood Search, GVNS), чији кораци су приказани Алгоритмом 4.

Алгоритам 4: Основна структура методе GVNS.

/ Иницијализација */*
Дефинисати низ околина N_k, $k = 1, 2, ..., k_{\max}$ које се користе за размрдавање;
Изабрати скуп околина N_l, $l = 1, 2, ..., l_{\max}$ које се користе за лок. претрагу;
Дефинисати критеријум заустављања;
На случајан начин изабрати почетно решење $x \in X$ и $\overline{x} \leftarrow x$;
понављај наредне кораке **док** није испуњен критеријум заустављања
1: Поставити $k \leftarrow 1$;
понављај наредне кораке **док** није $k > k_{\max}$
/ Размрдавање */*
На случајан начин изабрати тачку x' у околини $N_k(\overline{x})$;
/ Локална претрага */*
2: Поставити $l \leftarrow 1$;
понављај наредне кораке **док** није $l > l_{\max}$
/ Истраживање околине */*
Пронаћи најбоље решење x'' у околини $N_l(x')$;
/ Да ли се померити? */*
Ако $f(x'') < f(\overline{x})$ **онда**
Поставити $x' \leftarrow x''$, $f(\overline{x'}) \leftarrow f(x'')$ и **иди на** 2;
иначе
Поставити $l \leftarrow l + 1$;
крај
/ Да ли се померити? */*
Ако $f(x'') < f(\overline{x})$ **онда**
Поставити $\overline{x} \leftarrow x''$, $f(\overline{x}) \leftarrow f(x'')$ и **иди на** 1;
иначе
Поставити $k \leftarrow k + 1$;
крај
крај
крај
Стоп. Тачка $\overline{x}$ је апроксимација решења проблема.

2.1.5 Метода променљивих околина са декомпозицијом

Метода променљивих околина са декомпозицијом (енгл. Variable Neighborhood Decomposition Search, VNDS) је комбинација BVNS-а и идеје разбијања проблема на потпроблеме мањих димензија [36]. Наиме, за почетну

тачку x, сви атрибути (нпр. координате тачке) осим њих k, су фиксирани приликом локалне претраге. На тај начин се дефинишу модификоване околине $V_k(x)$, $k = 1, 2, ..., k_{max}$, где је $V_k(x) \subseteq N_k(x)$. Почетан корак методе је исти као и за BVNS, изабере се на случајан начин тачка x' из околине $N_1(x)$. Уместо претраге целе околине $N_1(x)$, сада се решава проблема налажења локалног минимума по једном (јер је на почетку $k = 1$) нефиксираном атрибуту који је случајно одабран. Нека је y скуп таквих атрибута (на почетку је то само један), а скуп фиксираних y'. Затим се враћа решење локалне претраге y'', генерише се ново решење као скуп фиксираних атрибута и добијеног y'' и пореди се са до тада најбољим решењем x. Уколико није постигнуто побољшање, прелази се на следећу околину ($k = 2$) и итеративно се понавља поступак са размрдавањем и локалном оптимизацијом док се не нађе боље решење од тренутног. Сада је простор $V_k(x)$ димензије 2 и решава се дводимензионалан проблем локалне претраге. Итеративно се понавља потупак за $k \geq 2$. Када је за неко k нађено боље решење $x'' = y'' \cup y'$, оно се прихвата ($x \leftarrow x''$) и претрага се даље наставља у околини $N_1(x)$. Критеријум заустављања се може дефинисати на различите начине, као максимално време извршавања методе, максималан број извршених итерација итд.

Алгоритам 5: Псеудо-шематски приказ методе VNDS.

/ Иницијализација */*
Дефинисати низ околина N_k, $k = 1, 2, ..., k_{\max}$;
Дефинисати критеријум заустављања;
На случајан начин изабрати почетно решење $x \in X$ и $\overline{x} \leftarrow x$;
понављај наредне кораке **док** није испуњен критеријум заустављања
1: Поставити $k \leftarrow 1$;
понављај наредне кораке **док** није $k > k_{\max}$
/ Размрдавање */*
На случајан начин изабрати тачку x' у околини $N_k(\overline{x})$;
Одредити скуп атрибута y решења x' који не карактеришу $\overline{x}$,
тј. $y = x' \backslash \overline{x}$;
/ Локална претрага */*
Применити неку методу локалног претраживања у простору решења за y и добијени оптимум означити са y', а са x'' одговарајуће решење за полазни проблем, $x'' = (x' \backslash \overline{x}) \cup y'$;
/ Да ли се померити? */*
Ако $f(x'') < f(\overline{x})$ **онда**

Поставити $\overline{x} \leftarrow x''$, $f(\overline{x}) \leftarrow f(x'')$ и **иди на** 1;

иначе

Поставити $k \leftarrow k + 1$;

крај

крај

крај

Стоп. Тачка $\overline{x}$ је апроксимација решења проблема.

Понекад се, осим улазног параметра k_{max}, код VDNS методе уводи још један параметар који означава максимално дозвољено време за извршавање потпроблема или максимална димензија потпроблема који решавамо. Ако би се тај параметар достигао током имплементације методе, поставили бисмо $k = 1$, тј. наставили бисмо претрагу на потпроблему најмање димензије и тиме потенцијално убрзали процес претраживања. У зависности од природе проблема, може се дефинисати да наредна околина буде $N_{k'}$, где је $k' = k + k_{step}$ (уместо $k = k + 1$), или $k = k_{min}$ (уместо $k = 1$). Структура методе VNDS приказана је Алгоритмом 5.

2.1.6 Закошена метода променљивих околина

Код проблема оптимизације који имају много локалних минимума, они су најчешће близу један другом и методе засноване на локалном претраживању релативно лако проналазе локалне минимуме који су блиски. Међутим, постоје и проблеми код којих се локални минимуми налазе врло далеко једни од других, у изолованим деловима простора, међусобно ограђеним пространим изврнутим "планинама". У том случају, пожељно је користити структуру околина које су већих димензија, да бисмо „покрили" потенцијалан глобални минимум груписан око локалних. Са повећањем величине околина повећава се и процесорско време неопходно за њихово претраживање. У граничном случају метода се претвара у вишеструко локално претраживање (енгл. Multistart search), које је мање ефикасно. Један од начина којим би се овај проблем могао превазићи је примена Закошене методе променљивих околина (енгл. Skewed Variable Neighborhood Search, SVNS) [18, 33]. У овој модификацији основне VNS методе, дозвољава се прелазак из једног локалног минимума $\overline{x}$ у други локални минимум x'' и у случају погоршања функције циља. Критеријум за прелазак из једног решења у друго користи комбинацију

вредности функције циља у тим решењима и њихову удаљеност. Нека је са $\rho(\overline{x}, x'')$ означена удаљеност решења $\overline{x}$ од x'', као нпр. Еуклидско растојање. Прелазак из решења $\overline{x}$ у x'' биће дозвољено уколико је $f(x'') - \alpha \cdot \rho(\overline{x}, x'') < f(\overline{x})$, где f означава вредност функције циља, а α је унапред задати позитиван параметар. Параметар α се бира експериментално тако да омогући претраживање околина далеко од x када је $f(x'')$ веће од $f(x)$. На пример, да би се избегло померање од x ка блиским решењима када је $\rho(x, x'')$ мало, може се изабрати велика вредност за параметар α. За $\alpha = 0$, SVNS метода се своди на методу BVNS.
Алгоритмом 6 дата је псеудо-шема за SVNS.

Алгоритам 6: Псеудо-шема методе SVNS.

/* *Иницијализација* */
Дефинисати низ околина N_k, $k = 1, 2, ..., k_{\max}$;
Дефинисати критеријум заустављања;
На случајан начин изабрати почетно решење $x \in X$ и $\overline{x} \leftarrow x$;
Одредити параметар α и метрику ρ;
понављај наредне кораке **док** није испуњен критеријум заустављања
1: Поставити $k \leftarrow 1$;
понављај наредне кораке **док** није $k > k_{\max}$
/* *Размрдавање* */
На случајан начин изабрати решење x' у околини $N_k(\overline{x})$;
/* *Локална претрага* */
Применити неку методу локалног претраживања у околини решења x';
Нека је x'' добијени локални минимум проблема;
/* *Побољшање или не?* */
Ако $f(x'') < f(\overline{x})$ **онда**
Поставити $\overline{x} \leftarrow x''$, $f(\overline{x}) \leftarrow f(x'')$ и **иди на** 1;
Ако $f(x'') - \alpha \cdot \rho(x'', \overline{x}) < f(\overline{x})$ **онда**
Поставити $\overline{x} \leftarrow x''$ и **иди на** 1;
иначе
Поставити $k \leftarrow k + 1$;
крај
крај
крај
Стоп. Тачка $\overline{x}$ је апроксимација решења проблема.

Параметар методе α се бира експериментално тако да омогући претраживање околина далеко од x када је $f(x'')$ веће од $f(x)$. На пример, да би се избегло померање од x ка блиским решењима када је $\rho(x, x'')$ мало, може се изабрати велика вредност за параметар α и супротно. Обично је потребно извршити прелиминарне експерименте у циљу одређивања адекватне вредности параметра α за конкретан проблем који се решава SVNS методом не би ли се пронашла одговарајућа формула за наведени параметар.

2.1.7 Паралелизација

Идеја паралелног извршавања алгоритма је проистекла из жеље за убрзањем процеса и смањење рачунарског времена које је потребно за решавање проблема, а такође и за повећањем квалитета истраживања простора у ком се налази решење (преко тзв. независних нити претраге). Неколико стратегија за паралелизацију VNS-а је предложено у литератури (погледати [12, 54]). Паралелизација се може постићи користећи програмски интерфејс, нпр. OpenMp, који служи за повезивање рачунарских језгара и њихове меморије, уз одређену стратегију (или више њих). У раду [12], предложено је неколико стратегија за паралелизацију. Једна од могућих стратегија је истраживање различитих околина VNS методе паралелно, тако што се кораци методе (размрдавање и локална претрага) извршавају у исто време на различитим процесорима анализирајући различите околине. У раду [32] извршена је паралелизација дефинишући процесоре по приоритетима, са једним надређеним и неколико подређених процесора за претраживање више околина. Надређени процесор води рачуна о критеријуму заустављања, ажурирању решења, доноси одлуке о усмеравању даље претраге, док подређени процесори извршавају делове VNS методе почевши од корака размрдавања. Више детаља о идеји паралелизације и начинима реализације се може наћи у [12, 54].

2.2 Метода променљивих околина у континуалној оптимизацији

Иако је првобитно служила за решавање проблема дискретне оптимизације, где и данас налази велику примену, у последњих десетак го-

дина метода промењљивих околина успешно је примењивана и на проблеме континуалне оптимизације. Први континуални проблем за који је предложена примена VNS-а је био *Веберов проблем* (енгл. Multisource Weber problem), у литератури познат и под називом *локацијско-алокацијски проблем* (енг. Location-alocation problem) [7, 8]. Код овог проблема, дато је n потрошача који се налазе на унапред познатим локацијама, а задатак је наћи позиције (у простору $\mathbb{R}^2$) m нових постројења који услужују поменуте потрошаче, тако да се минимизује укупна цена транспорта робе (и/или укупно време неопходно за опслуживање потрошача). Наредни проблем континуалне оптимизације на који је примењен VNS је уопштени проблем билинеарног програмирања (енгл. General bilinear programming problem) [31]. Иако оба ова проблема припадају континуалној оптимизацији, главни кораци за решавање су комбинаторне природе. На пример, за Веберов проблем, околина неке тачке (континуалног решења) се може дефинисати или премештањем објеката или реалокацијом потрошача. На овај начин укупан број решења је увек коначан. VNS метода у континуалној оптимизацији, без елемената дискретизације је први пут предложена у [55] где је разматрано решавање нелинеарног проблема без ограничења за дизајн расутог спектра полифазног радара (енгл. Spread spectrum radar polyphase code design problem). Данас се изворни VNS комбинује и са другим метахеуристичким методама, у циљу побољшања ефикасности алгоритма. Међутим, често се та комбинација (у стручној литертури позната и под називом *хибридизација*) ослања на карактеристике конкретног проблема, тј. сувише је комплексна и, за разлику од VNS-а, ретко се успешно може применити на више различитих проблема, иако су проблеми слични. У последњих неколико година, развијен је софтверски пакет GLOB за решавање нелинеарних проблема са интервалним ограничењима [45, 51], који је усавршен у раду [17] додавањем више различитих варијанти VNS-а које корисник има на располагању. За многе проблеме континуалне (као и комбинаторне) оптимизације, VNS метода се показала успешнијом од других метахеуристичких метода [34, 45, 55].

2.2.1 GLOB - VNS

Проблем глобалне континуалне оптимизације без ограничења се дефинише као:

$$\text{global}\min_{x\in X} f(x), \tag{2.1}$$

где је $f : \mathbb{R}^n \to \mathbb{R}$ непрекидна функција на компактном скупу X. У већини случајева из праксе, тражење глобалног минимума није лако спровести услед присуства често много локалних минимума, чији број може да расте експоненцијално у складу са порастом димензије проблема. Поред одређивања најмањег минимума међу свим локалним минимумима, неопходно је обезбедити и да рачунарско време потребно за решавање тог проблема буде што је могуће мање. Ипак, у општем случају, могуће је конструисати алгоритме који са неком дозвољеном грешком ϵ могу наћи решење, које је у том смислу апроксимација тачног решења.

Кључ GLOB-VNS-а је у дефинисању различитих околина које ће истражити простор решења. Такође, у раду [9] је показано да генерисање случајних тачака униформном расподелом у фази размрдавања, која је најчешћи избор у литератури, није увек и најбољи начин за случајан одабир тачке у конкретној околини. Управо ови параметри су истражени и довели до развоја софтвера GLOB-VNS [17, 52], који је конструисан у ширем смислу, тако што решава не само проблеме без ограничења, већ и проблеме са ограничењима.

Нека је N_k, $k = 1, 2, ..., k_{max}$ коначан скуп унапред дефинисаних околина и $N_k(x)$ скуп тачака које припадају околини N_k тачке x. Питања на која би требало обратити пажњу приликом дефинисања околина су следећа [34]:

а) Које карактеристике околина су обавезне да би метода могла да нађе глобални минимум (или решење „близу" глобалног минимума)?

б) Које особине околина су пожељне, с обзиром на искуство, у проналажењу решења?

в) Да ли би околине требало да буду угњеждене и ако не, како их уредити?

Прва два питања су најопштија и базирају се на жељи да се решење применом VNS методе нађе што брже, а да се притом, не знајући где се најбоље „долине" налазе, околине дефинишу тако да обиђу цео простор

X и да се не остави могућност да се вечито заглавимо у мањој „долини". Прецизније, желимо да цео скуп X буде садржан, у сваком тренутку, у унији свих околина $N_k(x)$, тј.

$$X \subseteq N_1(x) \cup N_2(x) \cup \cdots \cup N_{k_{max}}(x), \quad \forall x \in X.$$

Овај систем околина може бити формиран тако да буде угњежден, тј. да важи:

$$N_1(x) \subseteq N_2(x) \subseteq \cdots \subseteq N_{k_{max}}(x), X \subset N_{k_{max}}, \quad \forall x \in X.$$

Најчешће се угњеждене околине дефинишу на следећи начин:

$$N_k(x) = \{y | \rho(x, y) \leq \rho_k\} \tag{2.2}$$

где је $\rho(\cdot, \cdot)$ функција растојања, односно метрика у простору $\mathbb{R}^n \times \mathbb{R}^n$. Метрика је углавном неко l_p растојање између две тачке, где $1 \leq p \leq \infty$. Обично се узима да је $p = 1$, $p = 2$ или $p = \infty$. Дефинисањем метрике одређена је *структура околина* $\mathcal{G}$, а околина N_k је одређена геометријом $\mathcal{G}$ и бројем ρ_k. У раду [17] се користи неугњеждена структура околина:

$$N_k(x) = \{y | \rho_{k-1} \leq \rho(x, y) \leq \rho_k\} \tag{2.3}$$

за коју је показано да у случају бројних проблема континуалне оптимизације доводи до бољих резултата у односу на структуру угњеждених околина. Још један једноставан систем околина добија се компресијом скупа X око тачке x. Наиме, ако је X дефинисан као:

$$X = \{(x_1, x_2, ..., x_n \in \mathbb{R}^n) | a_i \leq x_i \leq b_i, a_i, b_i \in \mathbb{R}, i = 1, ..., n\}, \tag{2.4}$$

тада се околина око x^* може дефинисати као:

$$x_i^* - x_i \leq r_i(x_i^* - a_i) \tag{2.5}$$

$$x_i - x_i^* \leq r_i(b_i - x_i^*) \tag{2.6}$$

за неке $0 < r_i \leq 1$, $i = 1, ..., n$.
GLOB-VNS допушта да корисник сам одреди која ће се метрика користити или се метрике аутоматизовано током рада мењају по неком унапред дефинисаном редоследу. Радијуси ρ_k се мењају са повећањем ин-

декса k и вредности радијуса су или унапред дефинисане од стране корисника или се по аутоматизму израчунавају у процесу оптимизације. Поред дефинисања геометрије околина, јавља се и проблем начина дефинисања расподеле случајних бројева $\mathcal{P}$ која ће се користити за налажење случајне тачке x' из околине $N_k(x)$ у кораку размрдавања. У ову сврху, најчешће се у литератури користи униформна расподела. Сложеност израчунавања за генерисање тачке на случајан начин у $N_k(x)$, преко униформне расподеле, зависи од геометрије овог скупа. Одређивање случајне тачке са униформном расподелом у околини дефинисаној l_∞ нормом (n-димензионална коцка) је једноставно, али у случају да се користи нека друга норма, овај процес се усложњава. На пример, за генерисање случајних тачака униформно распоређених у оквиру јединичне кугле B могу се користити различити алгоритми. Један начин је метода пробе и грешке (енг. acceptance- rejection method) која најпре генерише случајну тачку, униформном расподелом у коцки $Q = [-1, 1]^n$, а затим се врши провера да ли се тачка налази унутар или изван B. У случају када је тачка ван B, она се одбацује и процес се понавља све док се не добије тачка која ће бити унутар кугле B. Ова метода је једноставна за имплементацију, али за веће димензије је доста неефикасна. Како су са порастом димензије n-димензионална коцка Q и унутар ње одговарајућа кугла B несразмерне, овај приступ није погодан за велике димензије. Алтернативно, могу се користити сферне координате како би се превазишао овај проблем, али у том случају алгоритам користи тригонометријске функције чије израчунавање значајно успорава алгоритам. Други, ефикаснији начин за генерисање случајних бројева преко униформне расподеле састоји се из два корака [18]:

1) Аренс - Дитер алгоритам ([1], [33]) се користи за брзо генерисање једнодимензионалне нормалне случајне промењљиве. Генерисањем n-димензионалног случајног вектора на овај начин, након нормирања, добија се, униформном расподелом, тачка на јединичној сфери.

2) Случајан полупречник r се генерише узимајући у обзир да је функција густине за r пропорционална површини сфере полупречника r, тј. Cr^{n-1}. Функција расподеле $F(x)$ и њен инверз су једноставни

за рачунање, те се r може добити као $r = F^{-1}(u)$, где је $u \in [0,1]$ униформно добијена случајна величина.

Структура GLOB-VNS методе приказана је Алгоритмом 7. Код ове методе, корисник може да изабере геометријске структуре $\mathcal{G}_l$ индуковане l_1, l_2 или l_∞ нормом у (2.2) и (2.3), као и за $\mathcal{P}_l$ униформну или хипергеометријску расподелу. Такође, корисник може да произвољно дефинише број и редослед парова геометрија околина и расподела $(\mathcal{G}_l, \mathcal{P}_l)$, $l = 1, ..., m$.

Алгоритам 7: Псеудо-шема методе GLOB-VNS.

/ Иницијализација */*
Изабрати парове $(\mathcal{G}_l, \mathcal{P}_l)$, $l = 1, ..., m$;
Поставити полупречнике ρ_i, $i = 1, ..., k_{max}$;
На случајан начин изабрати почетно решење $x \in X$ и $\overline{x} \leftarrow x$;
понављај наредне кораке **док** није испуњен критеријум заустављања
1: Поставити $l \leftarrow 1$;
понављај наредне кораке **док** није $l > m$
Формирати околине N_k, $k = 1, ..., k_{max}$ користећи геометријску структуру $\mathcal{G}_l$ и полупречник ρ_k;
Поставити $k \leftarrow 1$;
понављај наредне кораке **док** није $k > k_{\max}$
/ Размрдавање */*
Генерисати тачку y у околини $N_k(\overline{x})$ користећи расподелу $\mathcal{P}_l$;
/ Локална претрага */*
Применити неку методу локалне претраге у околини y како би се добио локални минимум y';
/ Да ли се померити? */*
Ако $f(y) < f(\overline{x})$ **онда**
Поставити $\overline{x} \leftarrow y$ и **иди на** 1;
иначе
Поставити $k \leftarrow k + 1$;
крај
крај
Поставити $l \leftarrow l + 1$;
крај
крај
Стоп. Тачка $\overline{x}$ је апроксимација решења проблема.

У случају процедуре локалне претраге, у литератури се може наћи више предлога. Из практичних разлога ефикасности је добро да на избору буду и методе погодне за оптимизацију недиференцијабилних функција, као и брже методе за оптимизацију глатких функција. У [4] предложен је метод области поверења (енг. trust region), док у GLOB-VNS пакету [17], корисник има избор шест различитих метода локалне претраге [18]:

- Нелдер - Мид (енгл. Nelder - Mead, NM) метода [57] спада у класу метода директне претраге које не користе (парцијалне) изводе функције, па се зато користе за локалну претрагу недиференцијабилних функција. Идеја методе је да се користи симплекс, полиедар одређен са $n+1$ тачком. У свакој од итерација, једна од његових тачака се одбацује и замењује новом, чиме се добија нови симплекс. NM у својој оригиналној верзији користи три пробна корака: рефлексију, експанзију и контракцију. Овим корацима се формирају три нове тачке, а једна од њих замењује најлошију тачку симплекса. Уколико ниједна од ових нових тачака нема мању вредност функције циља од најлошије, најлошијих n тачака се помера ка најбољој у том тренутку, тзв. *скупљањем* (енг. shrinking). Овај поступак се понавља док симплекс не постане довољно мали или се не испуни неки други критеријум заустављања.

- Хук - Џевис (енгл. Hooke-Jeeves, HJ) метода [41] састоји се од два корака. Идеја је да се у свакој итерацији одреде смерови у којима график функције има „брда" и „долине", а затим се „долине" следе до нове тачке. Поступак алгоритма започихње тако што се из тренутне тачке у свакој итерацији направе „кораци" дуж сваке од оса. Корак је успешан ако се вредност функције смањила, а неуспешан ако се вредност функције повећала или остала иста. За функцију са n променљивих, метода захтева (најмање) n смерова за претрагу. У зависности од тога како се мењају смерови претраге, неки од њих могу да стигну до локалног минимума релативно брзо, док неки смерови захтевају више времена за претрагу.

- Розенброк (енгл. Rosenbrock, RO) метода [60] у свакој итерацији врши претраживање дуж сваког од n ортогоналних праваца, где је n број променљивих у функцији циља. Померање тачке дуж једне

осе врши се за унапред задати корак. Уколико функција циља има мању вредност од вредности у претходној тачки, у следећем претраживању дуж истог правца вредност за коју се померамо се множи са познатим коефицијентом $\alpha > 1$. У случају да побољшање није постигнуто, у следећем претраживању дуж тог правца, вредност за коју се померамо се множи са $-\beta$, $0 < \beta < 1$. Овај корак итеративног процеса се понавља све док се у сваком правцу не појави бар једно успешно и једно неуспешно претраживање. У оригиналној верзији, у свакој наредној итерацији се обезбеђује да вектори праваца буду ортогонални познатим Грам-Шмитовим поступком ортогонализације.

- Метода најстрмијег спуста (енг. Steepest Descent, SD) полазећи од произвољне почетне тачке x_0 најпре одређује вредност градијента у тој тачки $\nabla F(x_0) = (\frac{\partial F}{\partial x_1}(x_0), \frac{\partial F}{\partial x_2}(x_0), \ldots, \frac{\partial F}{\partial x_n}(x_0))$. Градијент показује смер у ком функција најбрже расте у околини те тачке, док је смер најбржег опадања одређен антиградијентом $-\nabla F(x_0)$. Функција се претражује у том смеру, тј. решава се проблем једнодимензионалне оптимизације, не би ли се нашао коефицијент правца α_k. То се углавном постиже неком од тзв. метода *директне претраге*. Свака следећа тачка итеративног низа се рачуна по формули $x_{n+1} = x_n - \alpha_k \cdot \nabla F(x_n)$. Добијено решење се прослеђује као иницијално за наредну итерацију. Поступак се понавља док се не испуни неки критеријум заустављања (или више њих).

- Флечер - Пауел (Fletcher - Powell) метода [22] припада скупу тзв. квази-Њутнових метода. Поступак методе је следећи: када се у итеративном процесу дође до тачке x_k, $k = 0, 1, 2, ...$, најпре се одређује у њој вредност градијента и вредност у тачки x_0 одговарајуће апроксимације инверза Хесијан матрице у тачки x_k (матрице другог извода) функције $F : \mathbb{R}^n \to \mathbb{R}$ у k-тој итерацији, H_k. Затим се решава проблем минимизације функције једне променљиве, како би се одредили скалари α_k. Коефицијент α_k се одређује као тачка минимума (или апроксимација тачке минимума) функције $\phi(\alpha) = F(x_k + \alpha \cdot d_k)$, где је d_k решење једначине $d_k = -H_k \cdot \nabla F(x_k)$. Применом обрасца $x_{k+1} = x_k - \alpha \cdot H_k \nabla F(x_k)$, $k = 0, 1, ...$ налази се следећа тачка.

Матрица H_k одређује се из једнакости коју су предложили Флечер и Пауел:

$$H_{k+1} = H_k + \frac{S_k S_k^T}{S_k Y_k^T} - \frac{(H_k Y_k)(H_k Y_k)^T}{Y_k H_k Y_k}, k = 0, 1, ...$$

где је H_0 произвољна симетрична позитивно дефинитна матрица (обично $H_0 = I$), $S_k = x_{k+1} - x_k$ и $Y_k = \nabla F(x_{k+1}) - \nabla F(x_k)$.

- Флечер - Ривс (енг. Fletcher - Reeves) метода [23] припада класи метода коњугованих праваца у којој се не користи Хесијан већ само градијент функције. Свака следећа тачка итеративног низа се рачуна по формули $x_{k+1} = x_k + \alpha_k p_k$, где је $p_{k+1} = -\nabla f(x_{k+1}) + \beta_{k+1}^{FR} \cdot p_k$ и $\beta_{k+1}^{FR} = \frac{||\nabla f(x_{k+1})||^2}{||\nabla f(x_k)||^2}$. У сваком кораку се тражи коефицијент правца α_k неком методом претраге. Као почетна апроксимација за правац се узима $p_0 = -\nabla f(x_0)$. Итеративни поступак се понавља док се не испуни неки критеријум заустављања, обично када $||\nabla f(x_n)|| \leq \epsilon$.

Прве три методе локалне претраге не захтевају градијент функције и могу се користити и када функција циља није глатка. За преостале три методе неопходан је податак о градијенту функције који може бити или задат од стране корисника или израчунат неком од нумеричких метода.

2.2.2 Гаусовска метода променљивих околина за проблеме континуалне оптимизације

Уколико се за тренутак вратимо на полазни проблем (1.1) или (2.1), с обзиром да је број k_{max} различитих околина које се користе коначан, у највећем броју случајева није могуће, полазећи од тренутно најбоље тачке алгоритма VNS-а, достићи сваку тачку из скупа X. Због овога, постоји могућност да се не досегну области где се може налазити глобални минимум. Другим речима, ова идеја систематичних околина се може применити када проблем који се разматра нема ограничења типа (2.1), него је са интервалним ограничењима облика:

$$global \min_{x \in X} f(x) \tag{2.7}$$

где је $X = \{x \in \mathbb{R}^n | a_i \leq x_i \leq b_i, i = 1, 2, \ldots, n\}$. Параметри a_i и b_i су познати за конкретан проблем, односно зна се у ком распону, мањем или

већем, се налазе координате промењљиве. Сада је једноставније имплементирати идеје из поглавља (2.2.1) које се тичу дефинисања околина, односно покрити цео простор, сада простор интервалног ограничења X. Међутим, димензије околина не би требало да буду сувише велике, јер би то узроковало успорењу алгоритма претраге. Овде ће бити описана варијанта VNS-а која нема недостатак непокривања целог простора X у сваком тренутку итеративног процеса. Уместо дефинисања низа различитих физичких околина $N_1(x), ..., N_{k_{max}}(x)$ и процеса размрдавања у околини $N_k(x)$, може се узети да су све околине једнаке целом простору решења и дефинисати низ расподела $P_1(x), ..., P_{k_{max}}(x)$ које ће се користити у поступку размрдавања. Једноставности ради, може се узети за сваку $P_k(x)$ да је n-димензионална нормална расподела центрирана у x. Оваква варијанта VNS-а назива се *Гаусовска метода промењљивих околина* (Gaus-VNS) и први пут је предложена у раду [9]. Користећи овај приступ, теоријски је могуће „скочити" из текућег најбољег решења у било коју тачку целог простора решења. Тиме је могуће достићи до области где се налази глобални минимум полазећи од било које тачке, уколико паметно изаберемо матрицу коваријанси. Главна идеја Gaus-VNS-а је да се околине тачке x, $\{N_k(x)\}_{1\leq k\leq kmax}$, замене расподелама вероватноћа $\{P_k(x)\}_{1\leq k\leq kmax}$. У кораку размрдавања следећа случајна тачка ће се бирати користећи расподелу вероватноћа $P_k(x)$.

Уколико би се за $P_k(x)$ изабрала униформна расподела са носачем $N_k(x)$, тада би се добио класичан приступ VNS-а. За расподеле са неограниченим носачима природан избор је вишедимензионална Гаусова расподела случајних бројева. По Централној граничној теореми, низ случајних величина, под одређеним претпоставкама и околностима, грубо говорећи, конвергирају ка случајној промењљивој која има нормалну расподелу. Претпоставимо у даљем тексту да је $P_k(x)$ вишедимензионална Гаусова расподела са очекивањем x и матрицом коваријанси Σ_k. Другим речима, случајна тачка у кораку размрдавања се генерише n-димензионалним случајним вектором y са функцијом густине:

$$\phi(y) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma_k|}} e^{-\frac{1}{2}(y-x)^T\Sigma_k^{-1}(y-x)}, |\Sigma_k| = \det\Sigma_k.$$

Уколико желимо да имплементирамо методу, веома је важно да се користи ефикасан генератор случајних бројева са Гаусовом расподелом. Случајне величине са густином расподеле $P_k(x)$ се могу лако добити од

n независних вредности $z_1, ..., z_n$ добијених Гаусовом расподелом са математичким очекивањем 0 и дисперзијом 1, за коју постоје генератори псеудослучајних бројева. Један од примера је The Ziggurat Method [48]. Ако је $\Sigma_k = L_k L_k^T$ Холецки декомпозиција симетричне и позитивно дефинитне матрице Σ_k, онда је случајан вектор добијен као $y = x + L_k z$, где је $z = (z_1, ..., z_n)$, вектор са независним случајним величинама, са расподелом $\mathcal{N}(0, 1)$. Одавде, генерисање случајног вектора, користећи $P_k(x)$, се своди на рачунање Холецки декомпозиције матрице Σ_k, а затим генерисања n независних бројева користећи нормалну расподелу са очекивањем 0 и дисперзијом 1.
Поступак се додатно олакшава уколико се за матрицу коваријанси Σ_k узме умножак јединичне матрице I, тј.

$$\Sigma_k = \sigma^2 I, \quad k = 1, 2, ..., k_{\max}, \tag{2.8}$$

јер је у овом случају Холецки декомпозиција једноставно $\Sigma_k = (\sigma_k I)^T (\sigma_k I)$. Тада су координате z_i случајног вектора z једнодимензионалне независне Гаусове случајне величине са очекивањем 0 и дисперзијом σ_k.

Алгоритам 8: Псеудо-шема методе Gaus-VNS.

Изабрати скуп матрица коваријанси Σ_k, $k = 1, ..., k_{max}$;
На случајан начин изабрати почетно решење $x \in X$ и $\overline{x} \leftarrow x$;
понављај наредне кораке **док** није испуњен критеријум заустављања
1: Поставити $k \leftarrow 1$;
понављај наредне кораке **док** није $k > k_{\max}$
/* *Размрдавање* */
Генерисати y користећи Гаусову расподелу случајних бројева са очекивањем $\overline{x}$ и матрицом коваријанси Σ_k;
/* *Локална претрага* */
Применити неку методу локалног претраживања у околини решења y како би се добио локални минимум y' проблема;
/* *Да ли се померити?* */
Ако $f(y') < f(\overline{x})$ **онда**
Поставити $\overline{x} \leftarrow y'$, $f(\overline{x}) \leftarrow f(y')$ и **иди на** 1;
Поставити $k \leftarrow k + 1$;
крај
крај
Стоп. Тачка $\overline{x}$ је апроксимација решења проблема.

Након размрдавања и описаног начина добијања случајне тачке из целог простора претраге (X или $\mathbb{R}^n$), примењује се локална претрага у циљу добијања побољшаног решења. За добијање локалног минимума користи се нека од метода које су поменуте у претходном одељку (2.2.1). Псеудо-шемаски приказ методе Gaus-VNS-а дат је Алгоритмом 8.

3 Имплементација методе променљивих околина

У овом одељку је представљен начин имплементације методе променљивих околина, односно како су формиране фазе саме методе. За имплементацију VNS-а коришћено је програмско окружење MATLAB 2016b. При решавању нелинеарних проблема глобалне оптимизације, као мера квалитета методе најчешће се узима укупан број рачунања вредности функције циља до тренутка када је достигнуто оптимално (или најбоље) решење. Ова мера је погодна ради лакшег упоређивања резултата услед различитих перформанси рачунара на коме се експерименти изводе. Међутим, од улазних параметара попут критеријума заустављања или од избора методе локалне претраге зависи број позивања функције (и градијента) и дужина рада. У овом раду је, као мера квалитета, коришћен број позивања вредности функције циља.
За дефинисање структуре околина коришћене су l_p норме, где је $p = 1$, $p = 2$ и $p = \infty$. Околине су дефинисане формулама (2.2) и (2.3), где се формула (2.2) примењује уколико је параметар k достигао почетну вредност $k = 1$. Хеуристике које су коришћене за корак размрдавања су следеће:

X1: Скуп околина N_k, $k = 1, 2, ..., k_{max}$ су формиране користећи l_∞ норму. Случајна тачка $y \in N_k(x)$ у кораку размрдавања се добија тако што се прво генерише тачка z на јединичној l_∞ „сфери" униформном дистрибуцијом појединачних координата, а затим се, поново истом дистрибуцијом, на случајан начин изабере радијус из скупа $[0, \rho_k]$. Затим се скалирањем вектора z изабраним радијусом и транслирањем добије случајна тачка $y \in N_k(x)$;

X2: Скуп околина N_k, $k = 1, 2, ..., k_{max}$ су формиране користећи l_2 норму. Примењена је иста идеја као и за скуп околина које су добијене

хеуристиком X1, где се за одабир тачке z користила acception-rejection метода поменута у одељку (2.1);

X3: Скуп околина N_k, $k = 1, 2, ..., k_{max}$ су формиране користећи l_1 норму. Овде се случајна тачка $y \in N_k(x)$ бира кроз два корака: прво, случајна тачка $z = (z_1, z_2, ..., z_n)$ се генерише на l_1 „сфери" користећи специјалну, хипергеометријску дистрибуцију: z_1 координата се бира униформно из скупа $[-1, 1]$, док се за $k = 2, ..., n-1$, z_k бира униформно из скупа $[-A_k, A_k]$, где је $A_k = 1 - |z_1| - |z_2| - ... - |z_{k-1}|$, док последња кооордината z_n узима вредност A_n са произвољним (случајним) знаком. Затим се координате тачке z пермутују на случајан начин. На крају се на случајан начин, униформном дистрибуцијом, изабере радијус из скупа $[0, \rho_k]$, а затим се скалирањем вектора z изабраним радијусом и транслирањем добије случајна тачка $y \in N_k(x)$;
Хеуристика X3 је предложена у раду [17]. Наиме, искуство показује да се за велике димензије проблема процес претраге може значајно убрзати размрдавањем тако да се само неколико координата значајно разликују од тренутно најбољег решења x.

Хеуристике X1-X3 су модификоване у циљу њихове примене на начин дефинисан формулом (2.3).
Радијус ρ_k околине $N_k(x)$ се рачуна по формули $\rho_k = \frac{kd}{k_{max}}$, где је d максимална удаљеност тренутно најбоље тачке x од граница интервалног ограничења X (видети [17]).
Предложена VNS имплементација користи следеће процедуре локалне претраге:

- Нелдер-Мид метода, укратко објашњена у поглављу (2.1), је имплементирана на исти начин као у оригиналној верзији Нелдера и Мида у [57]. Формула за критеријум заустављања је:

$$Error = \frac{F_{max} - F_{min}}{\frac{|F_{max}|+|F_{min}|}{2} + 1} \tag{3.1}$$

- Модификација Нелдер-Мидове методе, чије су идеје преузете из [19]. Наиме, у фази скупљања (енгл. shrinking) се не мењају сва темена симплекса, већ само q оних чија је вредност функције циља

највећа, где је q број који у почетку узима вредност 0 и повећава се за 1 док год се није постигло побољшање у једној итерацији методе Нелдер-Мида. Када q достигне вредност димензије проблема, поставља се поново на 0.
Такође, још једна измена основне верзије Нелдер-Мидове методе састоји се у примени фазе рестартовања. Приликом извршења једног позива модификоване Нелдер-Мидове методе, решење се узима као почетна тачка новог позива, односно конструише се нови симплекс чије је једно теме управо последње добијено решење. Процес се итеративно понавља док не дође до унапред дефинисаног максималног броја рестартовања или ако нема побољшања.

- Метода најстрмијег спуста користи комбинацију идеја презентованих у [2], [66] и [67].
  За почетну итерацију се коефицијент правца α_0 тражи методом директне претраге. Она се заснива на комбинацији параболичке интерполације и идеје златног пресека, алгоритма чији су кораци представљени у [6]. Параболичку интерполацију примењујемо у свакој итерацији директне претраге да бисмо нашли апроксимацију тачке у којој се налази вредност следећег најбољег решења, док идеју златног пресека користимо као неку врсту „корекције", у случају да се није нашла тачка у тренутној околини која је најбоља, односно уколико се није остварило побољшање.
  Затим се у свакој наредној комбинују две формуле:

  1) Уколико је остатак при дељењу k са 4 један или два, онда је формула за коефицијент:

  $$\alpha_k = \frac{s_{k-1}s_{k-1}^T}{s_{k-1}y_{k-1}} \tag{3.2}$$

  где су $g_k = \nabla f(x_k)$, $s_{k-1} = x_k - x_{k-1}$ и $y_{k-1} = g_k - g_{k-1}$.

  2) Уколико остатак није један или два, онда се коефицијент рачуна по формули:

  $$\alpha_k = \frac{1}{\frac{||g_k||}{||s_{k-1}||} + \frac{1}{\alpha_{k-1}}} \tag{3.3}$$

За критеријум заустављања се користи оцена:

$$||g_k(x)||_\infty = \max_{1\leq k\leq n} |g_k(x)|. \tag{3.4}$$

- FRVJL метода (енгл. Fletcher-Reeves-Wei-Yao-Lui, FRWYL) је градијентна метода која користи Флечер-Ривсов коефицијент поправке правца:

$$\beta_k^{FR} = \frac{||g_{k+1}||^2}{||g_k||^2} \tag{3.5}$$

и коефицијент који су предложили Веи,Јао и Луи у [65]:

$$\beta_k^{WYL} = \frac{g_{k+1}^T(g_{k+1} - g_k(\frac{||g_{k+1}||}{||g_k||}))}{||g_k||^2}. \tag{3.6}$$

Свака наредна тачка итерације се рачуна као $x_{k+1} = x_k + \alpha_k d_k$. Коефицијент правца α_k се одређује директном претрагом [6], док се правац претраге добија из

$$d_{k+1} = -g_{k+1} + \beta_k^{FRWYL} d_k, \tag{3.7}$$

где је $\beta_k^{FRWYL} = \frac{1}{2}\beta_k^{FR} + \frac{1}{2}\beta_k^{WYL}$. На почетку је $d_0 = -g_0$.

- BFGS метода (енгл. Broyden–Fletcher–Goldfarb–Shanno,BFGS) је итеративна метода која припада класи квази-Њутнових метода за решавање проблема оптимизације. Користећи ознаке као код методе најстрмијег спуста, код BFGS методе се наредна тачка итеративног низа рачуна по формули $x_{k+1} = x_k + \lambda_k d_k$, где је d_k правац који се добија као решење једначине $B_k d_k = g_k$, где матрица B_k представља апроксимацију Хесијан матрице функције циља у тачки x_k. Формула за рачунање матрице B_k дата је формулом (изложеном у [56]):

$$B_{k+1} = \delta_k(B_k - \frac{B_k s_k^T s_k B_k}{s_k B_k s_k^T}) + \frac{y_k^T y_k}{y_k^T s_k^T}. \tag{3.8}$$

Број λ_k је коефицијент правца у k-тој итерацији који се добија директном претрагом, користећи квадратну интерполацију и методу златног пресека [6]. Уместо решавања система за d_k, апроксимира се заправо инверз Хесијана B_k функције циља у k-тој итерацији,

$H_k = B_k^{-1}$. Формула за H_k из [56] је:

$$H_{k+1} = \frac{1}{\delta_k}[H_k - \frac{H_k y_k s_k + s_k^T y_k^T H_k}{y_k^T s_k^T} + (\frac{\delta_k}{\gamma_k} + \frac{y_k^T H_k y_k}{y_k^T s_k^T})\frac{s_k^T s_k}{y_k^T s_k^T}], \tag{3.9}$$

где су $\gamma_k = \min\{1, \frac{y_k^T s_k^T}{||y_k||^2 + |s_k g_{k+1}|}\}$ и $\delta_k = \frac{n - \gamma_k \frac{||y_k||^2}{y_k^T s_k^T}}{n - \frac{||B_k s_k^T||^2}{s_k B_k s_k^T}}$.

Као параметар за критеријум заустављања узима се $||g(x_k)||_\infty$.

У MATLAB-у је имплементација извршена тако што је за сваку процедуру локалне претраге формирана посебна функција која ту идеју претраге уграђује у методу променљивих околина. Функције су назване на следећи начин:

- *VNSNM* - комбинација VNS-а и оригиналне методе Нелдер-Мида;
- *VNSBFGS* - комбинација VNS-а и квази Њутнове методе BFGS;
- *VNSFRWYL* - комбинација VNS-а и градијентних метода Fletcher-а, Reevs-а, Wei-а, Yao-а и Lui-а;
- *VNSSD* - комбинација VNS-а и методе најстрмијег спуста;
- *VNSRMNM* - комбинација VNS-а и модификоване Нелдер-Мидове методе.

Све поменуте методе се у командном прозору позивају са:

$$[broj\_poziva, y, x_0] = Ime\_metode(f, k_{\max}, region, m, max\_it, tolloc, varargin),$$

где је f функција циља (чији су аргументи, односно променљиве задате у векторском облику), region опсег променљивих задат у облику матрице $n \times 2$, где се у i-ту врсту матрице уписује, редом, најмање и највеће ограничење i-те координате променљиве, k_{max} максималан број околина одређене геометријске структуре (хеуристике), m је параметар којим одређујемо коју хеуристику (од горе поменутих) желимо да применимо. Уколико је $m = 1, m = 2$ или $m = 3$ примењује се одговарајућа хеуристика X1, X2 или X3. Уколико желимо да функцију позовемо за све три хеуристике па да се оне редом мењају када се достигне параметар

броја околина k_{max}, то се може учинити позивањем за вредност $m = 4$. Параметар *tolloc* је тачност локалне претраге. Број *max_it* је параметар максималног броја позива функције циља и програм прекида са извршавањем уколико након извршења локалне претраге, број позива функције постане већи од *max_it*. То је овде главни критеријум заустављања. Међутим, уколико је позната вредност минимума неке функције циља f, он се опционо може задати у оквиру структуре *varargin* типа cell-array, где је први аргумент структуре познати минимум, а други је критеријум заустављања којим дефинишемо која је тачност са којом желимо да наше крајње решење одступа од тачне, задате минималне вредности функције циља. Излазни параметри су вредност нађеног минимума y, укупан број позивања функције циља *br_poziva*, као и случајна тачка x_0 која је у опсегу одређеном са *region*.
Што се тиче рачунања градијентног вектора, сви парцијални изводи се апроксимирају формулом за нумеричко диференцирање тачности $O(h^2)$, тј. i-ти парцијалан извод у тачки $x = (x_1, x_2, ..., x_n)$ се рачуна по формули:

$$f'_{x_i}(x) = \frac{f(x_1, ..., x_{i-1}, x_i + h, x_{i+1}, ..., x_n) - f(x_1, ..., x_{i-1}, x_i - h, x_{i+1}, ..., x_n)}{2h},$$

где је корак h постављен на вредност $h = 0.0001$.
Међутим, није у интересу појединачно позивање наведених функција, већ тестирање просечних и најбољих резултата више таквих позива. За ту сврху се користе имплементиране функције:

- $av = averageNM(f, k_{\max}, region, m, max\_it, tolloc, k, varargin)$ - за VNSNM;
- $av = averageRMNM(f, k_{\max}, region, m, max\_it, tolloc, k, varargin)$ - за VNSRMNM;
- $av = average(f, k_{\max}, region, m, max\_it, tolloc, k, varargin)$ - за VNSBFGS, VNSSD i VNSFRWYL.

Овде улазни параметар k означава број позивања одговарајуће методе. За функцију *average*() корисник сам уноси коју локалну претрагу (односно скраћеницу исте) жели да примени и тестира. Као излазне аргументе ове функције враћају:

1) $av(1)$ - просечан број позивања функције циља у k извршавања;

2) $av(2)$ - просечан минимална вредност функције добијена у k извршавања;

3) $av(3)$ - најбржи резултат (најмањи број позивања функције циља) добијен у k извршавања;

4) $av(4)$ - број успешних извршавања, односно број оних извршавања који су довели до тачног решења f_{min} (уколико је познат) у околини дефинисаној критеријумом заустављања. Уколико минимум није познат, број успешних извршавања није излазни аргумент функције.

Такође, позивом наведених функција за тестирање се на излазу, у командном прозору, штампају случајне тачке од којих се полазило при узастопним позивима ових функција. Неки низови случајних тачака при позиву функција за тестирање за одређену тест функцију сачувани су у фајлу podaci.m.
Све тестиране функције се налазе у MATLAB фајлу funkcije.m или као засебно дефинисане функције у фајлу који је назван као и сама функција, као на пример Dikson.m, Rozenbrok.m итд. Такође, опсег променљивих за сваку од тест функција се налази у фајлу funkcije.m, где је матрица названа додавањем скраћенице за функцију на реч reg (на пример, regcv за Колвил функцију, regras10 за Растригин функцију димензије 10 итд.).

4 Експериментални резултати

У овом одељку представљена су поређења резултата добијених имплементацијом VNS алгоритма из одељка 3 са добијеним резултатима постојећих хеуристика из литературе. Најпре се врше поређења резултата са успешним VNS хеуристикама из литературе, а затим са резултатима других метахеуристика за ресавање проблема глобалне оптимизације. Као параметар поређења коришћен је број позивања функције циља.
Вредности улазних параметара који су дали најбоље резултате за GLOB-VNS дати су у [17]. Исте вредности параметара су примењене и за Gaus-VNS [18].
За вредност улазног параметра k_{max} је у резултате унесена она вредност параметра за коју је алгоритам довео до решења за најмање времена (у контексту броја позивања функције циља). У циљу поређења, преузети

су резулати из [64] и [4] и приказани су у овом раду.

4.1 Стандардне тест функције

За први део тестирања су узете глатке стандардне тест функције из литературе [4,37]. Ове функције су изабране из два разлога. Први је што су исте функције у литератури узете за тестирање и упоређивање резултата. Други разлог је тај што су карактеристике ових тест функција довољно разнолике да покривају велики број проблема који се јављају у глобалној оптимизацији. У скупу стандардних тест функција налазе се: функције које имају неколико изолованих локалних минимума попут Голдштајн и Прајс функције, функције са много нагомиланих локалних минимума попут Шуберт или Растригин функције, функције чији глобални минимум лежи у веома уском отвору (попут бунара) као што је случај са Исом функцијом или функције које имају облик врло узане долине (попут кањона) као што је случај са Розенброк функцијом [37].
У наставку је дат преглед стандардних тест функција и њихових основних карактеристика.

1) **Бранин функција (RC)**

- Број променљивих: 2.
- Дефиниција: $RC(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10.$
- Интервално ограничење: $5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15.$
- Број локалних минимума (поред глобалног): нема.
- Тачка глобалног минимума:
  $(x_1, x_2) = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475).$
- Вредност глобалног минимума: 0.397887.

2) **Исом функција (ES)**

- Број променљивих: 2.
- Дефиниција: $ES(x_1, x_2) = -\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}.$
- Интервално ограничење: $-10 \leq x_1 \leq 10,\ -10 \leq x_2 \leq 10.$

- Број локалних минимума (поред глобалног): много локалних минимума.
- Тачка глобалног минимума: $(x_1, x_2) = (\pi, \pi)$.
- Вредност глобалног минимума: -1.

3) **Голдштајн и Прајс функција (GP)**

- Број променљивих: 2.
- Дефиниција: $GP(x_1, x_2) = u \cdot v$, где је:
  $u = 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)$,
  $v = 30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)$.
- Интервално ограничење: $-2 \leq x_1 \leq 2,\ -2 \leq x_2 \leq 2$.
- Број локалних минимума (поред глобалног): 4 локална минимума.
- Тачка глобалног минимума: $(x_1, x_2) = (0, -1)$.
- Вредност глобалног минимума: 3.

4) **Растригин функција (RT)** [37]

- Број променљивих: 2.
- Дефиниција: $RT(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$.
- Интервално ограничење: $-1 \leq x_1 \leq 1,\ -1 \leq x_2 \leq 1$.
- Број локалних минимума (поред глобалног): много локалних минимума.
- Тачка глобалног минимума: $(x_1, x_2) = (0, 0)$.
- Вредност глобалног минимума: 0.

5) **Хамп функција (HM)**

- Број променљивих: 2.
- Дефиниција: $HM(x_1, x_2) = 1.0316285 + 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$.
- Интервално ограничење: $-5 \leq x_1 \leq 5,\ -5 \leq x_2 \leq 5$.

Табела 1: Таблица коефицијената за $H_{3,4}$

k	a_{kj}	c_k	p_{kj}
1	3.0 10.0 30.0	1.0	0.6890 0.1170 0.2673
2	0.1 10.0 35.0	1.2	0.4699 0.4387 0.7470
3	3.0 10.0 30.0	3.0	0.1091 0.8732 0.5547
4	0.1 10.0 35.0	3.2	0.0381 0.5743 0.8828

- Број локалних минимума (поред глобалног): нема.
- Тачка глобалног минимума: $(x_1, x_2) = (0.0898, -0.7126), (-0.0898, 0.7126)$..
- Вредност глобалног минимума: 0.

6) **Шуберт функција (SH)**

- Број променљивих: 2.
- Дефиниција: $SH(x_1, x_2) = (\Sigma_{k=1}^{5} k\cos(k + (k+1)x_1))(\Sigma_{k=1}^{5} k\cos(k + (k+1)x_2))$.
- Интервално ограничење: $-5 \leq x_1 \leq 5,\ -5 \leq x_2 \leq 5$.
- Број локалних минимума (поред глобалног): 742 локална минимума.
- Тачка глобалног минимума: 18 глобалних минимума.
- Вредност глобалног минимума: -186.7309.

7) **Хартман функција ($H_{3,4}$)**

- Број променљивих: 3.
- Дефиниција: $H_{3,4}(x_1, x_2) = \Sigma_{k=1}^{4} c_k e^{-\Sigma_{i=1}^{3} a_{ki}(x_i - p_{ki})^2}$.
- Интервално ограничење: $0 \leq x_j \leq 1,\ j = 1, 2, 3$.
- Број локалних минимума (поред глобалног): 4 локална минимума.
- Тачка глобалног минимума: $(x_1, x_2, x_3) = (0.114614, 0.555649, 0.852547)$.
- Вредност глобалног минимума: -3.86278.

8) **Колвил функција (CV)**

Табела 2: Таблица коефицијената за $S_{4,10}$

и	a_{ij}	c_i
1	4.0 4.0 4.0 4.0	0.1
2	1.0 1.0 1.0 1.0	0.2
3	8.0 8.0 8.0 8.0	0.2
4	6.0 6.0 6.0 6.0	0.4
5	3.0 7.0 3.0 7.0	0.4
6	2.0 9.0 2.0 9.0	0.6
7	5.0 5.0 3.0 3.0	0.3
8	8.0 1.0 8.0 1.0	0.7
9	6.0 2.0 6.0 2.0	0.5
10	7.0 3.6 7.0 3.6	0.5

- Број променљивих: 4.
- Дефиниција: $CV(x_1, x_2) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$.
- Интервално ограничење: $-10 \leq x_j \leq 10,\ j = 1, 2, 3, 4$.
- Број локалних минимума (поред глобалног): нема.
- Тачка глобалног минимума: $(x_1, x_2, x_3) = (1, 1, 1, 1)$.
- Вредност глобалног минимума: 0.

9) **Шекел функција ($S_{4,m}$)**

- Број променљивих: 4.
- Дефиниција: $S_{4,10}(x_1, x_2) = -\Sigma_{i=1}^{10}[\Sigma_{j=1}^{4}(x_j - a_{ij})^2 + c_i]^{-1}$.
- Интервално ограничење: $0 \leq x_j \leq 10,\ j = 1, 2, 3, 4$.
- Број локалних минимума (поред глобалног): 10 локалних минимума.
- Тачка глобалног минимума: $(x_1, x_2, x_3) = (4, 4, 4, 4)$.
- Вредност глобалног минимума: -10.5364.

10) **Хартман функција ($H_{6,4}$)**

- Број променљивих: 6.

- Дефиниција: $H_{6,4}(x_1, x_2) = \Sigma_{k=1}^{4} c_k e^{-\Sigma_{i=1}^{6} a_{ki}(x_i - p_{ki})^2}$.
- Интервално ограничење: $0 \leq x_j \leq 1,\ j = 1, 2, 3.$
- Број локалних минимума (поред глобалног): 6 локалних минимума.
- Тачка глобалног минимума:
  $x = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300)$.
- Вредност глобалног минимума: -3.32237.

Табела 3: Прва таблица коефицијената за $H_{6,4}$

i	a_{ij}	c_i
1	10.00 3.00 17.00 3.50 1.70 8.00	1.0
2	0.05 10.00 17.00 0.10 8.00 14.00	1.2
3	3.00 3.50 1.70 10.0 17.00 8.00	3.0
4	17.00 8.00 0.05 10.00 0.10 14.00	3.2

Табела 4: Друга таблица коефицијената за $H_{6,4}$

i	p_{ij}
1	0.1312 0.1696 0.5569 0.0124 0.8283 0.5886
2	0.2329 0.4135 0.8307 0.3736 0.1004 0.9991
3	0.2348 0.1451 0.3522 0.2883 0.3047 0.6650
4	0.4047 0.8828 0.8732 0.5743 0.1091 0.0381

11) **Гриванк функција (GR)**

- Број променљивих: 6.
- Дефиниција: $GR(x) = \Sigma_{j=1}^{6} \frac{x_j^2}{4000} - \Pi_{j=1}^{6} \cos(\frac{x_j}{\sqrt{j}}) + 1.$
- Интервално ограничење: $-1 \leq x_j \leq 1,\ j = 1, 2, ..., 6.$
- Број локалних минимума (поред глобалног): много локалних минимума.
- Тачка глобалног минимума: $(x_1, x_2, x_3, x_4, x_5, x_6) = (0, 0, 0, 0, 0, 0)$.
- Вредност глобалног минимума: 0.

12) **Диксон функција (DX)**

- Број променљивих: 10.
- Дефиниција: $DX(x) = (1 - x_1)^2 + (1 - x_{10})^2 + \Sigma_{j=1}^{9}(x_j^2 - x_{j+1})^2$.
- Интервално ограничење: $-10 \leq x_j \leq 10,\ j = 1, 2, ..., 10$.
- Број локалних минимума (поред глобалног): нема.
- Тачка глобалног минимума: $(x_1, ..., x_{10}) = (1, ..., 1)$.
- Вредност глобалног минимума: 0.

13) **Розенброк функције (R_n)**

- Број променљивих: 2,5,10.
- Дефиниција: $R_n(x) = \Sigma_{j=1}^{n-1}[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2]$.
- Интервално ограничење: $-5 \leq x_j \leq 10,\ j = 1, 2, ..., n$.
- Број локалних минимума (поред глобалног): нема.
- Тачка глобалног минимума: $(x_1, ..., x_n) = (1, ..., 1)$.
- Вредност глобалног минимума: 0.

14) **Закаров функције (Z_n)**

- Број променљивих: 2,5,10.
- Дефиниција: $\Sigma_{j=1}^{n} x_j^2 + (\Sigma_{j=1}^{n} 0.5 j x_j)^2 + (\Sigma_{j=1}^{n} 0.5 j x_j)^4$.
- Интервално ограничење: $-5 \leq x_j \leq 10, j = 1, 2, ..., n$.
- Број локалних минимума (поред глобалног): нема.
- Тачка глобалног минимума: $(x_1, ..., x_n) = (0, ..., 0)$.
- Вредност глобалног минимума: 0.

15) **Растригин функције (RA_n)**

- Број променљивих: $n = 1, 2, 3, ...$
- Дефиниција: $f(x) = 10n + \Sigma_{k=1}^{n}(x_k^2 - 10\cos(2\pi x_k))$.

- Интервално ограничење: $-5.12 \leq x_j \leq 5.12,\ j = 1, 2, ..., n.$
- Број локалних минимума (поред глобалног): 11^n.
- Тачка глобалног минимума: $(x_1, ..., x_n) = (0, ..., 0)$.
- Вредност глобалног минимума: 0.

За преглед резултата узета је рачунска сложеност, тј. укупан број рачунања вредности функције (који се узима у обзир и приликом тражења градијента функције циља) све док алгоритам не пређе дозвољен број позива функције циља или не пронађе апроксимацију глобалног минимума, уколико је он задат. Критеријум заустављања, уколико нам је позната најмања вредност функције циља f_{min}, је:

$$|f_{min} - y| \leq 10^{-4}|f_{min}| + 10^{-6}, \tag{4.1}$$

док је максималан број позива функције циља за све тест примере постављен на $4 \cdot 10^5$.
Програм је извршаван по 10 пута за сваку тест функцију и за крајњи резултат су узете просечне вредности добијене у свих 10 извршавања. Резултати су представљени у Табели 5 која је организована на следећи начин:

- У првој колони налази се одговарајућа ознака функције.
- У другој колони је димензија проблема тј. број променљивих тест функције.
- У трећој колони се налази вредност улазног параметра k_{max}.
- Наредне две колоне се односе на резултате познате из литературе: VNS-1 [4] и VNS-2 [64]. Резултати су представљени рачунском сложеношћу за добијање оптималног решења.
- Колона означена са „ЛОК. МИН." садржи информацију о томе која метода локалне претраге, од оних које су изложене у одељку 3, је коришћена за сваку тест функцију.
- Наредне две колоне садрже просечну рачунску сложеност алгоритама Glob-VNS и Gaus-VNS за добијање оптималних решења у 10 извршавања.

- Претпоследња колона садржи информацију која хеуристика (или више њих), од оних које су представљене у одељку 3, је коришћена за сваку тест функцију.

- Последња колона садржи просечну рачунску сложеност варијанте VNS-а представљене у одељку 3.

У табели су уписани најбољи просечни резултати методе VNS-а, као и за коју локалну претрагу, за коју хеуристику и за коју вредност k_{max} се постигао најмањи број позива функције циља.
Најбољи резултат, тј. најмањи број у табели, који представља рачунску сложеност за добијање оптималног решења, је истакнут.
Требало би напоменути да ови резултати нису директно упоредиви са алгоритмом представљеним у овом раду због другачијег критеријума заустављања и толеранције која се користила у одређеним деловима програма.

Табела 5: Експериментални резултати за стандардне тест функције

функ.	n	$k_{\max}$	VNS-1	VNS-2	ЛОК. МИН.	GLOB-VNS	Gaus-VNS	хеур.	VNS
RC	2	10	153	308	NM	131	112	X1	**78**
ES	2	10	167	-	NM	163	148	X1	**135**
GP	2	10	-	206	NM	260	**116**	X2	184
RT	2	10	246	-	$RMNM$	206	199	X1	**190**
HM	2	10	335	-	$FRWYL$	160	**80**	X2	147
SH	2	10	366	-	$RMNM$	382	591	X2	**264**
$H_{3,4}$	3	10	249	521	$RMNM$	246	**223**	X1	324
$H_{6,4}$	6	10	735	1244	SD	**397**	448	X2	561
CV	4	10	854	-	$BFGS$	669	**497**	X3	1119
$S_{4,10}$	4	5	590	988	$BFGS$	599	**399**	X3	857
GR	6	10	807	-	$RMNM$	135	126	X3	**75**
DX	10	10	2148	-	$BFGS$	1640	1576	X3	**1485**
R_2	2	10	556	-	$RMNM$	158	**125**	X1	235
R_5	5	5	**1120**	-	$BFGS$	1286	1308	X3	1495
R_{10}	10	10	2653	-	$BFGS$	**2357**	2561	X3	2581
Z_5	5	5	837	-	$BFGS$	728	461	X2	**234**
Z_{10}	10	10	1705	-	$FRWYL$	1142	**1010**	X3	1037
RA_{10}	10	10	-	-	SD	52471	85589	X3	**14083**
RA_{30}	30	10	-	-	SD	366950	599635	X1	**52548**
RA_{50}	50	10	-	-	SD	1334842	1504701	X1	**86863**

Као што се може видети из Табеле 5, VNS алгоритам предложен у овом раду је у 10 од наведених 20 тест функција дао најбољи резултат,

док је у неколико случајева релативно упоредив са GLOB-VNS-ом или Gaus-VNS-ом. Циљ ових тестова је био директно поређење наведених алгоритама, користећи исте или сличне параметре, ради добијања реалистичније слике. Видимо да су за најмање димензије најпогоднији локални претраживачи NM и $RMNM$ заједно са хеуристикама X1 и X2, док за димензије веће од 3 су доминантни $BFGS$ и SD, док је све присутнија хеуристика X3 у представљеним најбољим резултатима.

4.2 Резултати VNS методе и поређења са резултатима других метахеуристика на стандардним тест функцијама

У овом одељку приказано је поређење резултата предложене VNS методе са неколико познатих метахеуристика предложених у литератури за решавање проблема глобалне оптимизације, као што су: TS, GA, мрављи алгоритми, раштркана претрага (енгл. scatter search) [39], метахеуристике засноване на интелигенцији роја (енгл. swarm intelligence) и њихови хибриди. Изабране методе и њихови најбољи резултати за сваку појединачну функцију циља представљени су у Табели 6. У прве две колоне се налазе називи метода и њихове скраћене ознаке.
У трећој колони су дате референце на литературу из којих су преузети резултати. Како не постоји устаљена терминологија за имена метахеуристика на српском језику, оне су у табели наведене у оригиналу.
У последњој колони су наведене ознаке критеријума заустављања који су коришћени, за сваку методу.
Извршити директно поређење ових метода је јако тешко из више разлога. Један је већ споменути критеријум заустављања: различите методе користе различите формуле за мерење успешности методе, као и крај процеса претраге за решењем. Критеријуми заустављања у наведеним методама, чије ознаке су у последњој колони Табеле 6, су:

$$|\overline{f} - f_{min}| < 10^{-4} \cdot |f_{min}| + 10^{-6} \tag{4.2}$$

$$|\overline{f} - f_{min}| < 10^{-4} \cdot |f_{init}| + 10^{-6} \tag{4.3}$$

$$|\overline{f} - f_{min}| < 10^{-4} \cdot |f_{min}| + 10^{-4} \tag{4.4}$$

Табела 6: Метахеуристике за решавање проблема глобалне оптимизације

Метахеуристика	Ознака	Реф.	Заустављање
Genetic and Nelder-Mead	GNM	[61]	(4.2)
Continuous Reactive Tabu Search	CRTS	[3]	(4.1)
Swarm with Nelder-Mead	SNM	[68]	(4.2)
Continuous scatter search	CSS	[39]	(4.1)
Restarted modified Nelder-Mead	RMNM	[69]	(4.1)
Hybrid Continuous Interacting Ants	HCIAC	[62]	(4.3)
Directed Tabu Search with adaptive pattern search	DTS	[38]	(4.1)
Variable neighborhood search	VNS	овај рад	(4.1)

У овим формулама, $\overline{f}$ означава вредност функције циља добијену применом одговарајуће методе, f_{min} је унапред позната вредност глобалног минимума, а f_{init} је просечна вредност функције циља добијена полазећи од 100 случајно изабраних допустивих тачака. Критеријум (4.1) је најстрожији па је за његово достизање потребно извршити већи број израчунавања вредности функције циља. Уколико је позната вредност глобалног минимума, предложени VNS користи управо овај критеријум заустављања. Главни критеријум заустављања је максималан број позива функције циља, одосно све док број позива функције циља не постане већи од унапред дефинисаног параметра max_it.
У Табели 7 дат је преглед резултата свих 8 метода. Поређење је вршено на основу просечног броја израчунавања вредности функције циља, полазећи од 100 различитих почетних тачака добијених на случајан начин, до тренутка док се не пронађе апроксимација познатог оптималног решења f_{min} у околини дефинисаној критеријумом заустављања или док се не престигне максималан број позива функције циља. Неке од метода нису успеле да достигну минимум у свих 100 позива. У тим случајевима, у загради је записан укупан број успешних извршавања и просечан број позивања функција је наведен само за успешна извршавања.

Експериментални резултати показују да VNS у просеку постиже боље резултате у односу на осталих седам метода. Резултати који су истакнути у Табели 7 означавају најмање просечне рачунске сложености, док искошене вредности означавају друге по квалитету. У обзир се узимају само методе које су у свих 100 покретања дошле до резултата. VNS је у 3 од 9 случајева требало најмање просечног рачунања вредности

Табела 7: Поређења резултата 8 метахеуристика из литературе

Ф-је	*GNM*	*CRTS*	*SNM*	*CSS*	*RMNM*	*HCIAC*	*DTS*	*VNS*
BR	295	**38**	230	65	*60*	-	212	71
GP	259	*171*	304	**108**	69 (80)	34533	230	226
HT_3	492	513	436	-	**67**	-	438	*325*
HT_6	930	*750*	-	-	398 (50)	-	1787 (83)	**547**
RO_2	459	-	440	292	**224**	18747	254	*235*
RO_{10}	14563 (83)	-	*3303*	5847 (75)	5946 (95)	-	9037 (85)	**3104**
S_5	698 (85)	**664**	850	1197	912 (90)	17761	819 (75)	*825*
S_{10}	635 (85)	**693**	-	-	318 (75)	-	828 (52)	*860*
SH	*345*	-	753	762	275 (40)	-	274 (92)	**266**

функције циља, док је други по квалитету у четири случаја.

4.3 Неглатке тест функције

За испитивање успешности предложене методе на функцијама које нису глатке користила се модификована метода Нелдер-Мида описана у одељку 3. Тест функције преузете су из радова [20, 28]. Као критеријум заустављања за налажење довољно блиског решења оптималном је изабран критеријум (4.3), док је критеријум заустављања локалне претраге постављен на вредност 10^{-5}, осим за функције из 6), где су бољи резултати постигнути за критеријум 10^{-4}. С обзиром на то да у неким случајевима алгоритам неће достићи оптимално решење које ће нам бити познато у наредним тест примерима, као додатан критеријум заустављања коришћен је максималан број позива функције, односно алгоритам се прекида уколико број позива функције циља, након извршења локалне претраге, постане већи од дозвољене вредности. Неглатке функције које су узете у разматрање су:

1) Generalization of MAXQ

$$f(x) = \max_{1 \leq i \leq n} x_i^2,$$

$$f_{min} = 0.$$

2) Generalization of MXHILB

$$f(x) = \max_{1 \le i \le n} |\sum_{j=1}^{n} \frac{x_j}{i+j-1}|,$$
$$f_{min} = 0.$$

3) Chained LQ

$$f(x) = \sum_{i=1}^{n-1} \max\{-x_i - x_{i+1}, -x_i - x_{i+1} + x_i^2 + x_{i+1}^2 - 1\},$$
$$f_{min} = -(n-1)\sqrt{2}.$$

4) Chained CB3 I

$$f(x) = \sum_{i=1}^{n-1} \max\{x_i^4 + x_{i+1}^2, (2-x_i)^2 + (2-x_{i+1})^2, 2e^{-x_i+x_{i+1}}\},$$
$$f_{min} = 2(n-1).$$

5) Chained CB3 II

$$f(x) = \max\{\sum_{i=1}^{n-1} x_i^4 + x_{i+1}^2, \sum_{i=1}^{n-1}(2-x_i)^2 + (2-x_{i+1})^2, \sum_{i=1}^{n-1} 2e^{-x_i+x_{i+1}}\},$$
$$f_{min} = 2(n-1).$$

6) Radar polyphase code design function

$$f(x) = \max\{\phi_1(x), \phi_2(x), ..., \phi_{2m}(x)\},$$
$$X = \{(x_1, x_2, ..., x_n) \in \mathbb{R}^n | 0 \le x_i \le 2\pi, i = 1, ..., n\}, \text{ где је } m = 2n-1 \text{ и}$$
$$\phi_{2i-1} = \sum_{j=i}^{n} \cos(\sum_{k=|2i-j-1|+1}^{j} x_k),\ i = 1, ..., n,$$
$$\phi_{2i} = 0.5 + \sum_{j=i+1}^{n} \cos(\sum_{k=|2i-j|+1}^{j} x_k),\ i = 1, ..., n-1,$$
$$\phi_{m+i} = -\phi_i,\ i = 1, ..., m.$$

4.4 Резултати VNS методе на неглатким тест функцијама

У Табели 8 су приказани статистички подаци о понашању предложене модификоване Нелдер-Мидове методе при тестирању наведених неглатких функција. Табела 8 је организована на следећи начин:

- У првој колони су уписана имена функција које тестирамо;

- У другој колони је наведен број променљивих одговарајуће тест функције;

- У трећој и четвртој колони су наведени резултати метода из [19], које примењују VNS методу са локалном претрагом RNM (Нелдер-Мид са фазом рестартовања) и методу RMNME (модификована и рестартована Нелдер-Мидова метода са експанзијом величине симплекса). У свакој од тих колона се налазе информације о: просечном броју позивања функције циља у 10 извршавања, броју успеха у процесу налажења оптималног решења (у загради) и просечној вредности решења у 10 извршавања (уколико алгоритам није успео да пронађе одговарајући оптимум у свих 10 покушаја).

- У петој колони су уписани резултати модификоване Нелдер-Мидове методе (скр. MODNM) комбиноване са VNS-ом, имплементиране и представљене у овом раду. Колона је организована на исти начин као и претходне четири колоне.

- Последња колона садржи информацију о томе за коју од изложених хеуристика из одељка 3 је алгоритам довео до најбољег решења.

За првих пет тест функција је максималан број позива функције циља постављен на 30000, док је за последње две постављен, редом, на $2 \cdot 10^5$ и $1.2 \cdot 10^6$. Као решења до којих тежимо стићи алгоритмом у последња два тест примера су преузета решења из рада [55] која су дала најбољи резултат у 10 извршавања и она су уписана у колону f_{min}. За почетну тачку итеративног процеса у случају Radar функције је узета вредност $x_0 = (x_1, ..., x_n)$, $x_i = 1$, $i = 1, ..., n$ [55].
На основу резултата датих у Табели 8, може се видети да VNS у комбинацији са модификованом Нелдер-Мидовом методом извршава успешно свих 10 покушаја у 3 од првих 5 случајева, док у два случаја налази решење које у просеку не одступа значајно од оптималног. Предложена метода је успела да нађе за Radar odd функцију 6) за димензију $n = 7$ минимум који се поклапа на прве 4 децимале са решењем из [55], док за $n = 10$ не одступа значајно од решења које је у трећој колони Табеле 8.

Табела 8: Тестирање неглатких функција

f	n	f_{min}	VNS+RNM		RMNME		VNS+MODNM		H
MAXQ	10	0	5601(10)	-	5387(10)	-	2054(10)	-	X1
MXHILB	10	0	10873(10)	-	7583(10)	-	20492(5)	0.0003	X2
LQ	10	-12.7279	5277(10)	-	4669(3)	-12.7254	23916(5)	-12.7264	X3
CB3 I	10	18	8292(10)	-	6334(10)	-	9724(10)	-	X1
CB3 II	10	18	9775(10)	-	7975(9)	18.003	6351(10)	-	X1
Radar	7	0.4972	-	-	-	-	201841	0.4972	X2
Radar	10	0.4105	-	-	-	-	1205191	0.4449	X2

4.5 VNS метода за различите структуре околина

У Табели 9 приказани су резултати на основу којих се може закључити како се VNS понаша при одређеној локалној претрази у зависности од промене околина, односно хеуристике. За локалну претрагу се користио модификовани Нелдер-Мид (MODNM). Максималан број околина је за сва тестирања постављен на $k_{max} = 10$ осим за тестирање хеуристике X123 где је $k_{max} = 5$, да би се брже редом мењале хеуристике X1, X2 и X3. Максималан број итерација $5 \cdot 10^4$. Критеријум заустављања је критеријум (4.3). Поред поменутих хеуристика X1, X2 и X3 у одељку 3, анализиран је и утицај такође поменуте хибридне хеуристике X123. Анализа је спроведена на седам тест функција, од тога четири глатке и три неглатке. Резултати су представљени у Табели 9.

Табела 9: Поређења резултата при промени хеуристике

f	n	X1		X2		X3		X123	
$S_{4,5}$	4	17243 (10)		27135 (7)	-8.6231	37537 (4)	-7.0941	41436 (3)	-6.5844
$H_{6,4}$	6	1199 (10)		2206 (10)		2701 (10)		2671 (10)	
$S_{4,10}$	10	15558 (10)		38580 (4)	-7.2914	35381 (4)	-7.2913	33483 (5)	-7.8321
Z_{10}	10	6352 (10)		5552 (10)		6251 (10)		6328 (10)	
MXHILB	10	26292 (4)	0.0003	20492 (5)	0.0003	23464 (4)	0.0002	26433 (3)	0.0002
CB3 I	10	9724 (10)		11017 (10)		17124 (10)		16147 (9)	18.0018
CB3 II	10	7693 (10)		7329 (10)		7317 (10)		6351 (10)	

На основу резултата приказаних у Табели 9 можемо да закључимо да начин дефинисања хеуристике је важан фактор у конструисању алгоритма претраге. Видимо да хеуристика X1 постиже у већем броју случајева најбоље резултате, осим за функције Z_{10}, MXHILB и CB3 II, где су редом боље хеуристике X2, X3 и X123. Међутим, из Табеле 5 се такође може приметити и да за неке друге методе локалне претраге (BFGS и SD пре свега) за димензије $n = 10$ и $n = 30$ је најбоље резултате углавном дала хеуристика X3, тако да веза између локалне претраге и околине није занемарљива.

5 Закључак

У овом раду приказана је метода променљивих околина и њена примена на проблеме континуалне оптимизације за проблеме мањих и већих димензије. Показала се као идејно једноставна метода која остварује боље резултате од претходно конструисаних метахеуристика. Заједно са недетерминистичким приступом случајног одабира тачака и алгоритмима локалне претраживања, ова метода испуњава кључну идеју претраге - расејавање (диверзификацију) по простору у потрази за решењем и задржавање у околини где се решење може налазити. Диверзификација остварена уз помоћ вишедимензионалне Гаусове расподеле се у великом броју примера показала успешнијом у односу на случајан, униформан избор тачке у фази размрдавања, поготово за велике димензије. Варијанта VNS-а представљена у овом раду се показала успешнијом у односу на неке модификације VNS-а познатим из литературе и у односу на неке друге (мета)хеуристичке методе, узимајући у обзир број позивања функције циља.
При тестирању глатких функција, метода је просечно дала најбоље резултате користећи локалне претраге попут модификоване Нелдер Мидове методе (MODNM), квази Њутнове методе BFGS (Broyden-Fletcher-Goldfarb-Shanno) и методе најстрмијег спуста SD (steepest descent method), као и хеуристику X3, мада у неким случајевима су остале локалне претраге и геометријске структуре давала боље резултате за улазну функцију циља. Имплементација VNS-а и Нелдер-Мидове методе са модификованом фазом *скупљања* и додатом фазом *рестартовања* се показала корисном приликом минимизације неглатких функција.
Неки од праваца даљег развоја и унапређења добијених резултата могу бити:

- Имплементација Гаусове расподеле за добијање случајних тачака, као и тестирање неких других расподела;
- Хибридизација методе са другим хеуристикама, нпр. са Табу претраживањем;
- Развијање функција у процедуралном програмском језику (нпр. *C*-у) за ефикасно имплементирање методе у контексту процесорског времена;

- Паралелизација и извршавање на вишепроцесорским рачунарима.

Литература

[1] https://www.linkedin.com/pulse/variable-neighborhood-search-vrp-muthusamy-chelliah.

[2] J. Barzilai, J. M. Borwein. *Two point step size gradient methods.* IMA Journal of Numerical Analysis, 8(1): 141–148, 1988.

[3] R. Battiti, G. Tecchiolli. *The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization.* Annals of Operations Research, 63(2): 151–188, 1996.

[4] M. Bierlaire, M. Themans, N. Zufferey. *A heuristic for nonlinear global optimization.* INFORMS Journal on Computing, 22(1): 59–70, 2010.

[5] C. Blum, A. Roli. *Metaheuristics in combinatorial optimization: Overview and conceptual comparison.* ACM Computing Surveys (CSUR), 35(3): 268–308, 2003.

[6] R. P. Brent. *Algorithms for minimization without derivatives.* Englewood Cliffs, NJ: Prentice-Hall, Vol. 5, 1973.

[7] J. Brimberg, P. Hansen, N. Mladenović. *Improvement and comparison of heuristics for solving the uncapacitated multisource weber problem.* Operations Research, 48(3): 444–460, 2000.

[8] J. Brimberg, N. Mladenović. *A variable neighborhood algorithm for solving the continuous location-allocation problem.* Cahiers du GERAD, 1995.

[9] E. Carrizosa, M. Dražić, Z. Dražić, N. Mladenović. *Gaussian variable neighborhood search for continuous optimization.* Computers & Operations Research, 39(9): 2206–2213, 2012.

[10] L. D. Chambers. *The practical handbook of genetic algorithms: applications.* Chapman and Hall/CRC, 2000.

[11] D. Cvetković, M. Čangalović, V. Kovačević-Vujčić, D. Dugošija, S. Simić, J. Vuleta. *Kombinatorna optimizacija.* DOPIS, 1996.

[12] T. Davidović, T. G. Crainic. *Mpi parallelization of variable neighborhood search.* Electronic Notes in Discrete Mathematics, 39(2012): 241–248, 2012.

[13] T. Davidović, D. Teodorović, M. Šelmić. *Bee colony optimization part i: The algorithm overview.* Yugoslav Journal of Operations Research, 25(1): 33–56, 2014.

[14] L. Davis. *Handbook of genetic algorithms.* 1991.

[15] M. Dorigo, V. Maniezzo, A. Colorni. *Ant system: optimization by a colony of cooperating agents.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions, 26(1): 29–41, 1996.

[16] K. Dowsland. *Modern heuristic tehniques for combinatorial problems.* Blackwell Scientific Publications, Oxford, 1993.

[17] M. Dražić, V. Kovačević-Vujčić, M. Čangalović, N. Mladenović. *Glob - a new vns-based software for global optimization.* Nonconvex Optimization and its applications, 2006.

[18] Z. Dražić. *Modifikacije metode promenljivih okolina i njihove primene za problem rasporedjivanja prenosa datoteka.* Doktorska disertacija, Matematički fakultet, Univerzitet u Beogradu, 2014.

[19] Z. Dražić, M. Dražić, N. Mladenović, D. Urošević, Q. Zhao. *Continuous vns with modified nelder-mead for non-differentiable optimization.* IMA Journal of Management Mathematics, 27(1): 75–88, 2016.

[20] M. L. Dukić, Z. S. Dobrosavljević. *A method of a spread spectrum radar polyphase code design.* IEEE Journal on Selected Areas in Comm, 5(1990): 743–749, 1990.

[21] V. Černy. *A thermodynamical approach to the travelling salesman problem: an efficient simulated algorithm.* Journal of Optimization Theory and Applications, 45(1): 41–51, 1985.

[22] R. Fletcher, M. J. Powell. *A rapidly convergent descent method for minimization.* The Computer Journal, 6(2): 163–168, 1963.

[23] R. Fletcher, C. M. Reeves. *Function minimization by conjugate gradients.* The Computer Journal, 7(2): 149–154, 1964.

[24] M. Gendreau, J. Potvin. *Handbook of metaheuristics.* Springer, 2010.

[25] F. Glover. *A future path for integer programming and links to artificial intelligence.* Computer and Operations Research, 13(5): 533–549, 1986.

[26] F. Glover, E. Taillard, D. de Werra. *A user's guide to tabu search.* Annals of Operations Research, 41(1): 1–28, 1993.

[27] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison - Wesley, Reading, Mass, 1989.

[28] M. Haarala. *Large-scale nonsmooth optimization: Variable metric bundle method with limited memory.* Ph.D. thesis, Jyvaskyla, University of Jyvaskyla, 2004.

[29] P. Hansen, B. Jaumard. *Algorithms for the maximum satisfability problem.* RUTCOR Research Report, 44(4): 279–303, 1990.

[30] P. Hansen, N. Mladenović. *An introduction to variable neighborhood search in.* META-HEURISTICS, Advances and Trends in Local Search Paradigms for Optimization, (S. Voss et al. eds.), 1999.

[31] P. Hansen, N. Mladenović. *Variable neighborhood search: Principles and applications.* European journal of operational research, 130(3): 449–467, 2001.

[32] P. Hansen, N. Mladenović, M. Gendreau, T. G. Crainic. *Cooperative parallel variable neighborhood search for the p-median.* Journal of Heuristics, 10(3): 293–314, 2004.

[33] P. Hansen, N. Mladenović, B. Jaumard, A. Parreira. *Variable neighborhood search for weighted maximum satisfability problem.* Les Cahiers du GERAD, 2000.

[34] P. Hansen, N. Mladenović, Q. Groupe d'retudes et de recherche en analyse des decisions Montreal. *A tutorial on variable neighborhood search.* Le Casiers du GERAD, 2003.

[35] P. Hansen, N. Mladenović, J. A. M. Perez. *Variable neighborhood search: methods and applications.* 4OR, 6(4): 319–360, 2008.

[36] P. Hansen, N. Mladenović, D. Perez-Britos. *Variable neighborhood decomposition search.* Journal of Heuristics, 7(4): 335–350, 2001.

[37] A. R. Hedar, M. Fukushima. *Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization.* Optimization Methods and Software, 17(5): 891–912, 2002.

[38] A. R. Hedar, M. Fukushima. *Tabu search directed by direct search methods for nonlinear global optimization.* European Journal of Operational Research, 170(2): 329–349, 2006.

[39] F. Herrera, M. Lozano, D. Molina. *Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies.* European Journal of Operational Research, 169(2): 450–476, 2006.

[40] J. Holland. *Adaptation in Natural and Artificial Systems.* 1975.

[41] R. Hooke, T. A. Jeeves. *Direct search solution of numerical and statistical problems.* Journal of the ACM, 8(2): 212–229, 1961.

[42] S. Kirkpatrick, C. Gellat, M. Vecchi. *Optimization by simulated annealing.* Science, 220(4598): 671–680, 1983.

[43] D. E. Knuth. *Postscript about np-hard problems.* ACM Special Interest Group on Algorithms and Computation Theory, 6(2): 15–16, 1974.

[44] G. A. Kochenberger, et al. *Handbook of metaheuristics.* Springer, 2003.

[45] V. Kovačević-Vujčić, M. Čangalović, M. Dražić, N. Mladenović. *Vns-based heuristics for continuous global optimization.* Le Thi Hoai An, Pham Dinh Thao (Eds.), Modelling, Computation and Optimization in Information Systems and Management Sciences. Hermes Sciences Publishing, 2004.

[46] P. Lučić, D. Teodorović. *Bee system: modelling combinatorial optimization transportation engineering problems by swarm intelligence in.* Preprints of the TRISTAN IV triennial symposium on trasportation analysis, Sao Miguel, Azores Islands, Portugal,, 2001.

[47] P. Lučić, D. Teodorović. *Computing with bees: attacking complex trans- portation engineering problems.* International Journal on Artificial Intelli- gence Tools, 12(3): 375–394, 2003.

[48] G. Marsaglia, W. W. Tsang. *The ziggurat method for generating random variables.* Journal of Statistical Software, 5(8): 1–7, 2000.

[49] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller. *Equation of state calculation by fast computing machines.* Journal of Chem. Phys., 21(6): 1087–1091, 1953.

[50] N. Mladenović. *A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization applications.* In Optimization Days, 1995.

[51] N. Mladenović, M. Dražić, M. Čangalović, V. Kovačević-Vujčić. *Variable neighborhood search in global optimization.* In: Mladenović, N., Dugošija, Đ(Eds.), Proceedings of XXX Symposium on Operations Research, Herceg Novi, 2003.

[52] N. Mladenović, M. Dražić, M. Čangalović, V. Kovačević-Vujčić. *General variable neighborhood search for the continuous optimization.* European Journal of Operational Research, 191(3): 753–770, 2008.

[53] N. Mladenović, P. Hansen. *Variable neighborhood search.* Computers and Operations Research, 24(11): 1097–1100, 1997.

[54] N. Mladenović, P. Hansen, J. A. M. Perez. *Parallel variable neighborhood search.* Groupe d'etudes et de recherche en analyse des decisions, 2004.

[55] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, M. Čangalović. *Solving spread spectrum radar polyphase code design problem by tabu search and variable neighborhood search.* European Journal of Operational Research, 151(2): 389–399, 2003.

[56] A. Neculai. *A double parameter scaled bfgs method for unconstrained optimization.* Journal of Computational and Applied Mathematics, 332(2018): 26–44, 2018.

[57] J. A. Nelder, R. Mead. *A simplex method for function minimization.* The computer journal, 7(4): 308–313, 1965.

[58] I. H. Osman, G. Laporte. *Metaheuristics: A bibliography.* Annals of Operations Research, 63(5): 511–623, 1996.

[59] C. Ribeiro, P. Hansen. *Essays and surveys in metaheuristics.* Kluwer Academic Publishers, 2002.

[60] H. Rosenbrock. *An automatic method for finding the greatest or least value of a function.* The Computer Journal, 3(3): 175–184, 1960.

[61] P. Siarry, R. Chelouah. *Genetic and nelder-mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions.* European Journal of Operational Research, 148(2): 335–348, 2003.

[62] P. Siarry, J. Dreo. *Hybrid continuous interacting ant colony aimed at enhanced global optimization.* Algorithmic Operations Research, 2(1): 52–64, 2007.

[63] E. G. Talbi. *Metaheuristics: From design to implementation.* Wiley Publishing, July 2009.

[64] M. D. Toksari, E. Guner. *Solving the unconstrained optimization problem by a variable neighborhood search.* Journal of Mathematical Analysis and Applications, 328(2): 1178–1187, 2007.

[65] Z. Wei, S. Yao, L. Liu. *The convergence properties of some new conjugate gradient methods.* Applied Mathematics and Computation, 183(2): 1341–1350, 2006.

[66] G. Yuan. *A conjugate gradient method for unconstrained optimization problems.* International Journal of Mathematics and Mathematical Sciences, 2009.

[67] Y. Yuan. *A new stepsize for the steepest descent method.* Journal of Computational Mathematics, 24(2): 149–156, 2006.

[68] E. Zahara, S. K. S. Fan, Y. C. Liang. *A genetic algorithm and a particle swarm optimizer hybridized with nelder-mead simplex search.* Computers & Industrial Engineering, 50(4): 401–425, 2006.

[69] Q. H. Zhao, D. Urošević, N. Mladenović, P. Hansen. *A restarted and modified simplex search for unconstrained optimization.* Computers & Operations Research, 36(12): 3263–3271, 2009.