Универзитет у Београду
Математички факултет

Милош Шошић

Хеуристички приступ решавању проблема минималног кашњења

мастер рад

Београд

2014.

Ментор: **проф. др Зорица Станимировић**
Математички факултет у Београду

Чланови комисије: **проф. др Миодраг Живковић**
Математички факултет у Београду

доц. др Мирослав Марић
Математички факултет у Београду

Датум одбране: _____

Хеуристички приступ решавању проблема минималног кашњења

Апстракт:

Проблем минималног кашњења представља варијацију проблема трговачког путника, где је циљ минимизовати суму времена потребног да се посети сваки од чворова, тј. суму кашњења. Овај проблем има широку практичну примену, која укључује дистрибуцију робе корисницима, логистику у кризним ситуацијама, распоређивање послова итд. Циљ овог рада је развој хеуристичке методе за решавање проблема минималног кашњења, заснованој на хибридизацији варијанте методе променљивих околина са методом симулираног каљења, чији су елементи прилагођени карактеристикама разматраног проблема. Имплементација овог хибридног алгоритма је тестирана на јавно доступним инстанцама за овај проблем и резултати су упоређени са тренутним најбољим решењима или оптималним где су позната.

Кључне речи: Проблем минималног кашњења, метода променљивих околина, симулирано каљење, хибридизација

апстракт енглески

Садржај

1 Увод

Оптимизација представља скуп математичких метода за решавање тешких математички дефинисаних проблема у многим областима као што је физика, биологија, медицина, економија и др. Други термин за оптимизацију је математичко програмирање. Основни делови сваког проблема који се решава оптимизацијом су функција циља, која одражава квалитет решења, променљиве одлучивања и ограничења проблема. Задатак оптимизације је одредити променљиве одлучивања тако да се минимизује (или максимизује) функција циља при задатим ограничењима.

Детерминистички оптимизациони проблем има формулацију:

$$min(f(x)|x \in X, X \subseteq S) \tag{1.1}$$

где је f реална функција циља, S простор решења а X скуп допустивих решења. Ако је S коначан или пребројиво бесконачан скуп, тада је оптимизација комбинаторна а ако је $S = \mathcal{R}^n$ тада је континуална. Решење x^* је оптимално ако важи:

$$f(x^*) < f(x), \forall x \in S \tag{1.2}$$

Тачни решавач оптимизационог проблема налази оптимално решење, уз доказ оптималности, или доказује да такво решење не постоји, тј. $S = \emptyset$.

За многе оптимизационе проблеме кардиналност скупа S расте експоненцијално са порастом димензије проблема па је практично немогуће наћи оптимално решење тачним решавачем за догледно време. Ти проблеми углавном припадају класи NP-тешких проблема за коју се не зна ниједан алгоритам полиномијалне сложености који би решио проблем који јој припада. Зато се користе *хеуристике*, алгоритми који дају приближна решења без гаранције оптималности али за кратко време извршавања. Због ових добрих особина, хеуристике имају широку примену у решавању разних оптимизационих проблема, посебно за инстанце великих димензија [1].

Метахеуристике су приближне методе за решавање тешких проблема комбинаторне оптимизације где хеуристике нису успеле да дају квалитетна решења или није постигнута тражена ефикасност. Ове методе користе неку унутрашњу хеуристику и пружају јој додатне информације како би се пронашло оптимално решење на ефикаснији начин од саме хеуристике.

У поглављу 2 се описује метода променљивих околина, у поглављу 3 метода симулирано каљење. У поглављу 4 описује се проблем минималног кашњења а у

поглављу 5 хибридни алогоритам. Поглавље 6 садржи опис тестирања имплементације алгоритма хибридизације и добијене резултате.

2 Метода променљивих околина

Методе комбинаторне оптимизације које користе локалну претрагу долазе до бољих решења итеративно примењујући локалне промене над тренутно најбољим решењем све док се не достигне локални оптимум. Ове локалне промене припадају околини $N(x)$ тренутног решења x. Код оваквих метода може се доћи до стања када алгоритам остане у неком локалном и не достигне глобални оптимум, тј. оптимално решење проблема. Ово стање се тада решава на различите начине механизмом који се назива пертурбација.

Метода променљивих околина (VNS[1]) је метахеуристика представљена у [2], која се базира на методама локалне претраге, где се користи систематска промена структуре околина како би се побољшало тренутно решење у процесу локалне претраге и пертурбације. Ова метода је осмишљена за приближно решавање проблема комбинаторне оптимизације али је временом проширена и корисити се при решавању проблема целобројног, мешовитог целобројног, нелинеарног програмирања и других. Стога, неке од области примене VNS методе су локацијски проблеми, проблеми распоређивања и упаривања, проблеми рутирања возила, рачунарске мреже, и многи други (за детаљнији преглед области примене, погледати [1]). Следи детаљнији опис VNS метахеуристике [1].

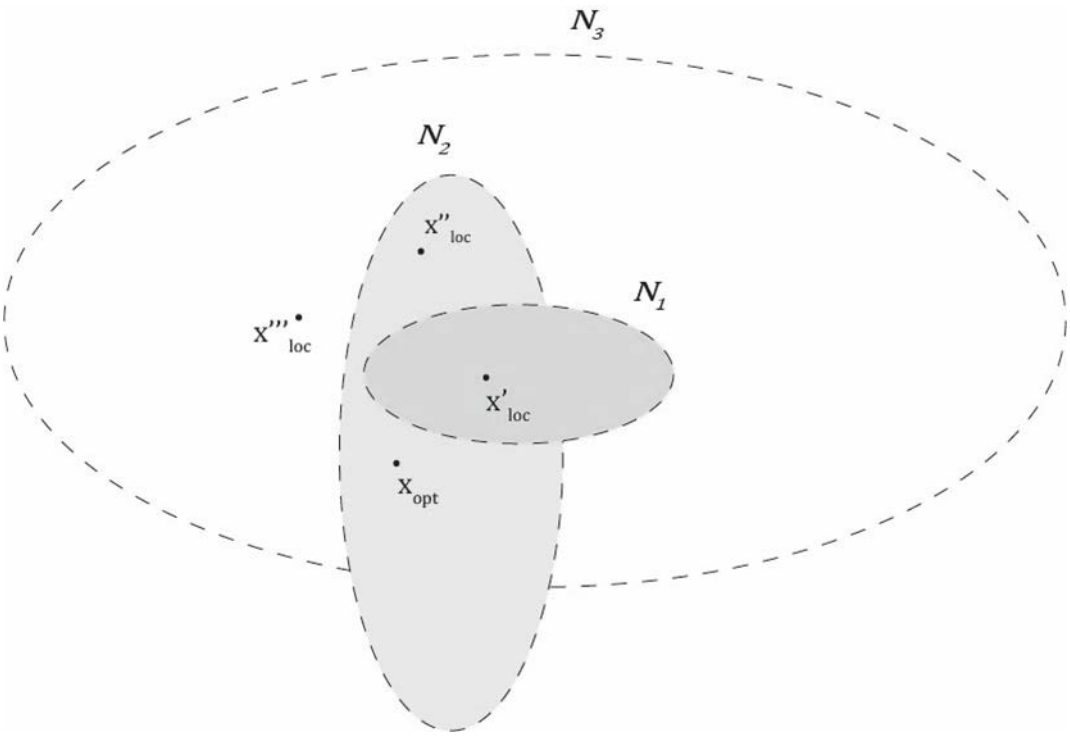
Нека је S дати простор решења и $f: S \to R$ функција циља за дати проблем. Нека су $N_k(k = 1, \dots, k_{max})$ пажљиво одабране структуре околина а $N_k(x)$ скуп решења из k-те околине решења x, где $x \in S$. Означимо оптимално решење са x_{opt} (глобални минимум). У околини N_k, x' је локални минимум ако не постоји $x \in N_k(x')$ тако да важи $f(x) < f(x')$. Да би различите околине биле добро дефинисане, треба да важи следеће:

1) Локални минимум једне околине не мора бити локални минимум друге околине
2) Глобални минимум је локални минимум свих могућих околина
3) У многим проблемима важи, локални минимуми различитих околина су релативно близу.

Последња претпоставка подразумева да локални минимум често пружа корисне информације о глобалном минимуму. На пример, може се десити да неке од променљивих узимају исте вредности у оба решења али како се не може утврдити које

[1] енг. *Variable Neighborhood Search*

су те променљиве у току извршавања алгоритма, примењује се пажљива претрага околине локалног минимума у потрази за бољим решењем. Пример структура околина може се видети на слици 1.



Слика 1. Пример структура околина где N_1, N_2, N_3 представљају околине, x'_{loc}, x''_{loc}, x'''_{loc} респективно њихове локалне оптимуме а x_{opt} глобални оптимум.

Основни VNS алгоритам састоји се из две фазе које се извршавају у свакој итерацији. Прва фаза је „размрдавање“ где се прави случајни потез у тренутној околини након чега следи друга фаза, локална претрага која налази локални минимум за текућу околину.

Процедура $\boldsymbol{VNS}()$

1. **иницијализација:** конструисати почетно решење x, избор структура околина $N_k(k \leftarrow 1, \dots, k_{max})$, изабрати критеријум заустављања
2. **понављати:** док се не испуни критеријум заустављања
3. $k \leftarrow 1$
4. **понављати:** док $k \neq k_{max}$
5. **размрдавање:** случајним потезом у околини $N_k(x)$ бира се тачка x'
6. ***локална прет рага:*** *претражује се околина $N_k(x')$ тачке x' и налази локални оптимум x''*
7. **промена решења:** ако је $f(x'') < f(x)$ тада $x \leftarrow x''$ и $k \leftarrow 1$, иначе $k \leftarrow k + 1$
8. **излаз:** x

Почетно решење може бити било које допустиво решење, обично добијено неким похлепним или алгоритмом са случајним одабиром променљивих одлучивања. Критеријум заустављања може бити утрошено процесорско време, максимални број итерација, максимални број итерација без побољшања решења итд. Локална претрага у овом алгоритму може подразумевати претрагу за најбољим решењем у тренутној околини (енг. *best-improvement*) или претрагу за првим бољим решењем (енг. *first-improvement)*. Уместо локалне претраге може се користити и било која друга s-хеуристика.

Једна значајна процедура локалне претраге са променљивим околинама је метода променљивог спуста (VND – енг. *Variable Neighborhood Descent*) дата следећим алгоритмом.

Процедура $\boldsymbol{VND}(\boldsymbol{x})$

1. **иницијализација:** избор структура $N_k(k \leftarrow 1, \dots, k_{max})$
2. $k \leftarrow 1$
3. **понављати:** док $k \neq k_{max}$
4. **локална претрага:** претражује се околина $N_k(x)$ тачке x и налази локални оптимум x'
5. **промена решења:** ако је $f(x') < f(x)$ тада $x \leftarrow x'$ и $k \leftarrow 1$
   иначе $k \leftarrow k + 1$
6. **излаз:** x

Ако се ова метода користи у другој фази алгоритма VNS, добија се генерализовани VNS (GVNS – енг. *General VNS*) који се за неке проблеме комбинаторне оптимизације истакао као најефикаснија метода за приближно решавање.

Процедура $\boldsymbol{GVNS}()$

1. **иницијализација :** конструисати почетно решење x, избор структура околина $N_k(k \leftarrow 1, \dots, k_{max})$, изабрати критеријум заустављања
2. **понављати:** док се не испуни критеријум заустаљања
3. $k \leftarrow 1$
4. **понављати:** док $k \neq k_{max}$
5. **размрдавање:** случајним потезом у $N_k(x)$ бира се x'
6. **локална претрага:** $x'' \leftarrow VND(x')$
7. **промена решења:** ако је $f(x'') < f(x)$ тада $x \leftarrow x''$ и $k \leftarrow 1$,
   иначе $k \leftarrow k + 1$
8. **излаз:** x

где се у процедури VND користе исте структуре околина као из дела иницијализације алгоритма GVNS.

3 Симулирано каљење

Симулирано каљење (SA – енг. *Simulated Annealing*) је веома широко проучавана метахеуристика која припада групи метода локалне претраге. Користи се при решавању дискретних и у мањој мери континуалних оптимизационих проблема. Главна одлика ове методе је то што дозвољава решењу да постане лошије како би се извршила пертурбација и избегла прерана конвергенција алгоритма ка неком локалном оптимуму.

Мотивација за ову метахеуристику је проистекла из процеса каљења метала у металургији. Метал се загрева и контролисано хлади како би се достигла најбоља конфигурација кристала и избегли дефекти. На овај начин, ако је погодно изабрана шема хлађења, метал који се на крају добије има побољшану структуру веза међу атомима и повољне особине за даљу обраду.

За дати проблем комбинаторне оптимизације, нека је S простор решења, $f: S \rightarrow R$ функција циља и нека је са $N(x)$ дефинисана околина тачке $x \in S$. Треба пронаћи x' тако да важи $f(x') \leq f(x)$ за свако $x \in S$. Симулирано каљење у свакој итерацији упоређује два решења, тренутно, x, и решење из његове околине, x', добијено на случајан начин. Уколико је x' боље решење, тренутно решење постаје x', а ако је лошије, x' се прихвата са одређеном вероватноћом која зависи од параметра алгоритма T. Овај параметар представља тренутну температуру процеса каљења и смањује се у току алгоритма према унапред одређеној шеми хлађења која подразумева скуп температура t_k $(k = 1, \dots, n)$. За свако t_k, бира се број итерација M_k на тој температури. Најзначајнији део алгоритма је вероватноћа прихватања решења x' која се израчунава на следећи начин:

$$P(\omega) = \begin{cases} 1, & f(x') - f(x) < 0 \\ e^{-\frac{f(x') - f(x)}{T}}, & f(x') - f(x) \geq 0 \end{cases} \tag{3.1}$$

где је ω догађај прихватања решења x'.

Процедура $SA()$

1. **иницијализација:** изабрати почетно решење, критеријум заустављања, шему хлађења t_k и M_k
2. $k \leftarrow 1$
3. **понављати:** док се не испуни критеријум заустављања

4. $m \leftarrow 0$

5. понављати: док $m \neq M_k$

6. изабрати $x' \in N(x)$ на случајан начин

7. $\Delta_{x,x'} \leftarrow f(x') - f(x)$

8. промена решења: ако је $\Delta_{x,x'} < 0$ онда $x \leftarrow x'$,
иначе $x \leftarrow x'$ са вероватноћом $e^{-\frac{\Delta_{x,x'}}{T}}$

9. $m \leftarrow m + 1$

10. $k \leftarrow k + 1$

11. излаз: x

где критеријум заустављања може бити као и код VNS алгоритма.

При већој температрути већа је и вероватноћа прихватања лошијег решења а како температура опада, та вероватноћа постаје све мања. На тај начин се на почетку избегава прерана конвергенција методе и омогућава претрага различитих ланаца решења док се на крају налази локални минимум прихватањем искључиво бољих решења. Успех методе у великој мери зависи од шеме хлађења па ову шему треба пажљиво конструисати.

Више о симулираном каљењу и доказу конвергенције методе ка глобалном решењу може се наћи у [3].

4 Проблем минималног кашњења

Нека је $G = (V, A)$ комплетан граф, где је $V = \{0, \dots, n\}$ скуп чворова, тј. локација које треба посетити, а $A = \{(i, j) \mid i, j \in V, i \neq j\}$ скуп грана графа G, где је уз сваку грану познато времене потребно за пут преко те гране. Постоји један истакнути чвор (обично чвор 0) који представља почетни чвор са којег обилазак почиње. Нека је $l(i)$ кашњење до i-тог чвора које се израчунава као потребно време да се стигне од истакнутог до чвора i. Циљ проблема минималног кашњења (MLP – енг. *Minimum Latency Problem*) је пронаћи Хамилтонов пут који минимизује суму $\sum_{i=0}^{n} l(i)$. У овом раду, сваки обилазак графа почиње са чвором 0 и завршава се када обиђе осталих n чворова. MLP је у литератури познат и по другим именима (енг. *Traveling Repairman Problem, Delivery Man Problem, Cumulative Traveling Salesman Problem, School Bus Driver Problem*).

MLP има широку примену у пракси, посебно у дистрибуцији робе корисницима, логистици у кризним ситуацијама, распоређивању послова итд. Код проблема путујућег мајстора, познате су локације клијената и мајстора, као и времена пута између клијената и потребна времена за сервисирање сваког клијента. Потребно је пронаћи пут којим мајстор обилази клијенте тако да је њихово укупно време чекања минимално. Сличан је проблем курира, где курир треба да пронађе пут који минимизује укупно време чекања на доставу пошиљке. Ови проблеми су клијентски орјентисани проблеми рутирања јер функција циља даје предност минимизацији времена чекања клијената у односу на дужину пута возила.

Код распоређивања послова на машини[2] јавља се овај проблем у следећем облику. Постоји скуп послова које машина треба да обави као и време које је потребно да се машина поново подеси да ради посао j након завршеног посла i. Ово време представља време пута од чвора i до чвора j. Треба пронаћи пермутацију задатих послова тако да се минимизује потребно време за завршетак свих послова. [4]

4.1 Математичка формулација

Нека је $G = (V, A)$ комплетан граф као у претходном делу и нека је дат Хамилтонов пут, тј. пермутација чворова где је чвор 0 на првом месту, који представља

[2] енг. *Machine Scheduling Context*

неко решење проблема. Нека је $C = (c_{ij})$ ненегативна матрица трошкова где c_{ij} одговара трошку гране од чвора i до чвора j. Кашњење $l([i])$[3], које одговара чвору на i-тој позицији у датом путу, може се израчунати као $l([i]) = l([i-1]) + c_{[i][j]}$, где $[0] = 0$ и $l([0]) = l(0) = 0$. Функција циља је представљена сумом $L = \sum_{i=0}^{n} l([i])$, а како важи

$l([1]) = l([0]) + c_{[0][1]}$

$l([2]) = l([1]) + c_{[1][2]} = c_{[0][1]} + c_{[1][2]}$

…

$l([n]) = l([n-1]) + c_{[n-1][n]} = c_{[0][1]} + c_{[1][2]} + \cdots + c_{[n-1][n]}$

тако је

$$L = \sum_{i=0}^{n} l([i]) = nc_{[0][1]} + (n-1)c_{[1][2]} + \cdots + 2c_{[\mathrm{n}-2][\mathrm{n}-1]} + c_{[\mathrm{n}-1][\mathrm{n}]}.$$

Код проблема курира и путујућег мајстора, вредност c_{ij} чине време пута t_{ij} и време сервисирања s_{ij}, тј.:

$$c_{ij} = \begin{cases} t_{0j}, & \text{за } i = 0,\ j = 1, \dots, n \\ s_i + t_{ij}, & \text{за } i = 1, \dots, n,\ j = 1, \dots, n,\ i \neq j \end{cases}$$

У овом раду, решава се проблем где је матрица C симетрична, тј. дужина пута између два чвора је иста у оба смера.

У свакој пермутацији чворова, Хамилтонов пут почиње од чвора 0 и обилази остале чворове по реду у којем је дата пермутација. Ако грана (i, j) спаја k-тог са $k+1$-им чланом пермутације, тада је допринос те гране функцији циља $(n-k)c_{[k][k+1]} = (n-k)c_{ij}$.

Уводимо следеће променљиве одлучивања:

$$x_i^{(k)} = \begin{cases} 1, & \text{ако је чвор } i \text{ на позицији } k \\ 0, & \text{иначе} \end{cases}$$

$$y_{ij}^{(k)} = \begin{cases} 1, & \text{ако је чвор } i \text{ на позицији } k \text{ а чвор } j \text{ на позицији } k+1 \\ 0, & \text{иначе} \end{cases}$$

Променљиве $x_i^{(1)}, (i = 1, \dots, n)$ се користе у функцији циља при израчунавању кашњења од чвора 0 до по-зиције 1, тј. чвора i, а променљиве $y_{ij}^{(k)}, (i = 1, \dots, n,\ j = 1, \dots, n,\ i \neq j,\ k = 1, \dots, n-1)$ за израчунавање кашњења осталих чворова.

[3] $[i]$ *представља i-ти чвор у датом Хамилтоновом путу*

Следи целобројна линеарна формулација проблема минималног кашњења на основу [4]:

$$\min z = n\sum_{i=1}^{n} c_{0i}x_i^{(1)} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}(n-k)c_{ij}y_{ij}^{(k)} \tag{4.1}$$

при ограничењима:

$$\sum_{k=1}^{n} x_i^{(k)} = 1, \qquad i = 1,2,\dots,n \tag{4.2}$$

$$\sum_{i=1}^{n} x_i^{(k)} = 1, \qquad k = 1,2,\dots,n \tag{4.3}$$

$$\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ij}^{(k)} = x_i^{(k)}, \qquad i = 1,2,\dots,n-1,\ \ k = 1,2,\dots,n-1 \tag{4.4}$$

$$\sum_{\substack{j=1\\ j\neq i}}^{n-1} y_{ji}^{(k)} = x_i^{(k+1)}, \qquad i = 1,\dots,n,\ \ k = 1,2,\dots,n-1 \tag{4.5}$$

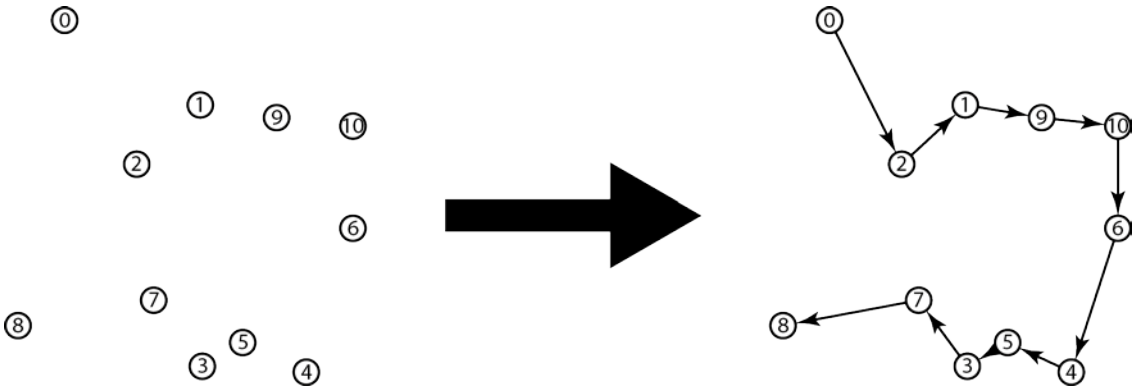
$$x_i^k \in \{0,1\}, \qquad i = 1,\dots,n,\ \ k = 1,\dots,n \tag{4.6}$$

$$y_{ij}^{(k)} \in \{0,1\}, \qquad i = 1,\dots,n,\ \ j = 1,\dots,n,\ \ j \neq i,\ \ k = 1,\dots,n-1 \tag{4.7}$$

Ограничење (4.2) представља услов да се сваки чвор налази на тачно једној позицији у пермутацији, (4.3) је услов да се на свакој позицији налази тачно један чвор, (4.4) ограничава да са сваке позиције води само једна грана ка следећем чвору а (4.5) да на сваку позицију само једна грана води са претходног. Ограничења (4.6) и (4.7) су бинарна ограничења променљивих одлучивања.

4.2 Пример

Нека је дата инстанца проблема минималног кашњења са 11 чворова. На слици 2 може се видети распоред чворова и оптималан пут за ту инстанцу.



Слика 2. Пример инстанце са 11 чворова и оптималан пут

4.3 Претходна решавања

Како проблем минималног кашњења припада класи NP-тешких проблема [5], тачни решавачи су примењиви само на инстанцама одређене величне. Први такав алгоритам [6], био је енумерациони, заснован на формулацији нелинеарног целобројног програмирања користећи Лагранжову релаксацију за добијање доњих ограничења. У овом раду решене су инстанце са мање од 30 чворова. Следи [7] у којем је предложена формулација целобројног програмирања а алгоритам за решавање је гранање са ограничавањем (енг. *branch-and-bound*) користећи матроидне структуре проблема за добијање доњих ограничења. Решене су инстанце до 60 чворова. Касније, у [8] је изложена формулација целобројног програмирања која користи предности везе са проблемом линеарног поретка а представљен је алгоритам одсецања равни који користи валидне неједнакости у тој формулацији. Два рада, [9] и [10], представила су *branch-cut-and-price* приступ којим су решене инстанце до 107 чворова. Ово је тренутно једини тачни алгоритам који решава инстанце до ових димензија.

Проблем минималног кашњења је решаван и приближним алгоритмима. Први овакав алгоритам [11] имао је апроксимациони фактор 144. Тренутно најбољи приближни алгоритам [12] има апроксимациони фактор 3.59.

Не постоји пуно радова у којима је MLP решаван помоћу метахеуристика. Један од њих је [13], где је коришћена метахеуристика која спаја GRASP методу и методу променљивих околина. За проблем минималног кашњења са профитом, развијена је табу претрага у [14]. Варијација проблема рутирања возила [15] може да се прилагоди тако да решава MLP. У том раду, описан је меметски алгоритам као и ефикасна процедура провере новог потеза у околини са $O(1)$ операција. Тренутно најзначајнији рад у којем се користи метахеуристика за решавање MLP-а је [16] у којем се спајају метода GRASP, итеративна локална претрага и варијација методе променљивих околина. Метахеуристика предложена у [16] достиже оптимална решења за инстанце

до 107 чворова где су оптимална решења позната, а за веће димензије постиже најбоља решења тренутно позната.

У неким од поменутих радова, нпр. [16], уместо Хамилтоновог пута треба пронаћи Хамилтонов циклус, тј. после обиласка свих чворова одлази се до чвора 0.

5 Хибридини алгоритам за проблем минималног кашњења

Идеја хибридних метахеуристика постоји већ дуже времена иако је термин „хибридизација“ прихваћен тек у последњих десетак година [17]. Под овим термином подразумева се комбиновање различитих метахеуристка у сложенији алгоритам. Успешна хибридизација представља спој добрих особина метахеуристика које се користе у алгоритму, како би се постигла квалитетнија решења за дати проблем или ефикаснији алгоритам.

За проблем минималног кашњења хибридни алгоритам који ће бити представљен користи методу променљивих околина као основни алгоритам са измењеном фазом локалне претраге. Ова фаза се састоји из комбиновања методе променљивог спуста са методом симулираног каљења. Идеја је да у почетку дође до доброг решења користећи методу променљивог спуста као локалну претрагу а онда уместо ње се користи метода симулираног каљења да би се дошло до квалитетнијег решења. Алгоритам је дат псеудокодом:

Процедура $\boldsymbol{VNS\_SA}()$

1. **иницијализација :** конструисати почетно решење x, избор структура околина $N_k(k \leftarrow 1, \dots, k_{max})$, изабрати критеријум заустављања, максимални број итерација MAX_{ITER}, параметар α, шему хлађења t_k i M_k

2. $iter \leftarrow 1$

3. **понављати:** док $iter < MAX\_ITER$

4. $k \leftarrow 1$

5. **понављати:** док $k \neq k_{max}$

6. **размрдавање:** случајним потезом у околини $N_k(x)$ бира се тачка x'

7. **локална претрага:** ако је $iter < \alpha$ тада $x'' \leftarrow VND(x')$,
иначе $x'' \leftarrow SA(x')$

8. **промена решења:** ако је $f(x'') < f(x)$ тада $x \leftarrow x''$ i $k \leftarrow 1$,
иначе $k \leftarrow k + 1$

9. $iter \leftarrow iter + 1$

10. **излаз:** x

У овом алгоритму се при позивању процедуре SA не конструише изнова почетно решење већ се за њено почетно узима решење x'.

Параметар MAX_{ITER} зависи од дате инстанце и износи $(\text{број чворова})/4$. Параметар α представља број итерација после којег треба започети примењивање симулираног каљења уместо методе VND. Емпиријски је закључено да добра вредност за $\propto$ износи $MAX_{ITER}/2$. Ова вредност доноси добар најбољи однос квалитета решења и брзине извршавања.

5.1 Кодирање и простор решења

Решење за MLP је Хамилтонов пут па се решење природно кодира низом бројева. Ови бројеви представљају индексе чворова у том путу а њихов редослед одговара редоследу обиласка чворова. Простор решења S је скуп свих пермутација бројева из скупа $\{0, \dots, n\}$ које почињу бројем 0 (јер је први чвор фиксиран). Пример једног решења за инстанцу MLP-а где важи $n = 10$ дат је на слици 3.

0	7	8	5	1	3	9	2	10	6	4

Слика 3. Пример кодирања решења за инстанцу где $n = 10$.

Нотација за представљање руте решења биће, за пример са слике 3, $x = [0, 7, 8, 5, 1, 3, 9, 2, 10, 6, 4]$.

5.2 Конструкција почетног решења

Почетно решење у фази иницијализације алгоритма VNS је похлепна хеуристика (NNH – енг. *Nearest Neighbor Heuristic*) која конструише руту чвор по чвор. Почиње се од чвора 0 и сваки наредни чвор је онај који има најкраћи пут од последњег чвора у тренутном путу. Овај алгоритам је дат следећим псеудокодом:

Процедура $\boldsymbol{NNH}()$

1. **иницијализација:** рута $x \leftarrow [0]$

2. $k \leftarrow 1$

3. **понављати:** док $k \leq n$

4. рути x се додаје чвор c, од преосталих, који је најближи последњем чвору у рути

8. $k \leftarrow k + 1$

9. **излаз:** x

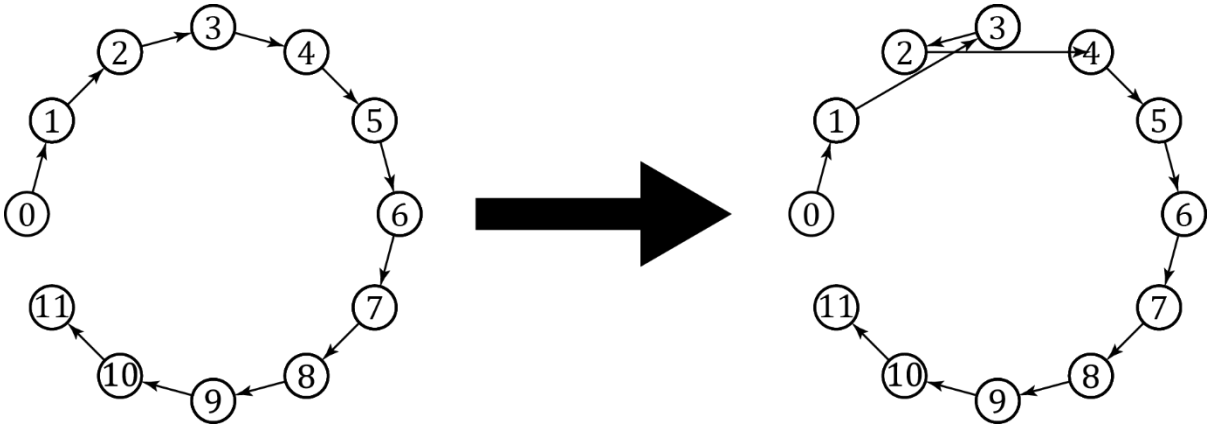
Овај алгоритам даје боље почетно решење од избора руте на случајан начин па се зато користи у овој имплементацији.

5.3 Структуре околине

Како би метода променљивих околина постизала квалитетна решења, потребно је пажљиво одабрати скуп структура околина. У овом раду те структуре су добро познате околине из литературе за проблем трговачког путника, које оправдавају овај избор квалитетом решења. Решење је у околини другог решења ако се од њега разликује за једну операцију која дефинше ту околину. Ове операције индукују редом околине N_1, N_2, N_3, N_4 и N_5 и биће изложене по величини околина које конструишу, почевши од најмање.

5.3.1 $SwapTwo$

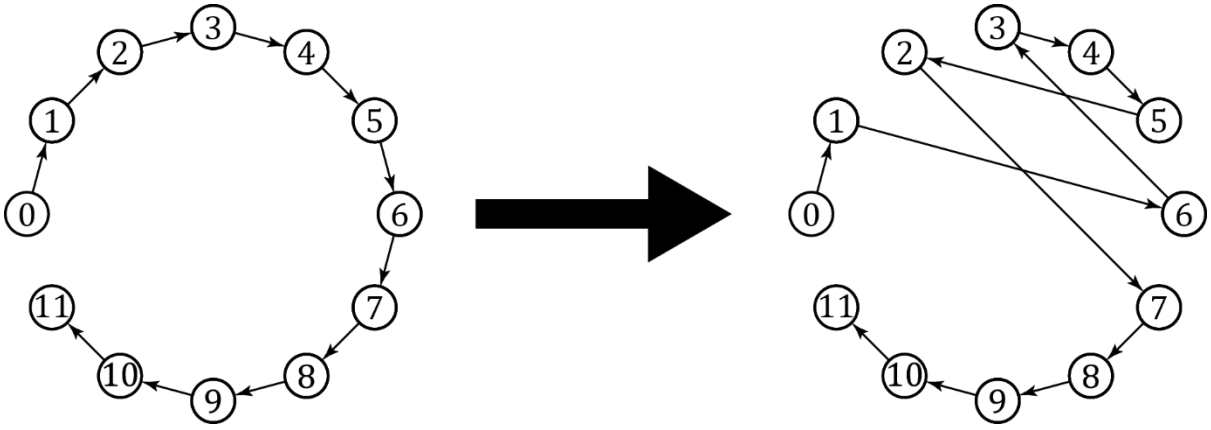
Процедура $SwapTwo$ замењује места два суседна чвора у датом путу решења. Како би се претражила цела $SwapTwo$ околина решења потребно је извршити $O(n)$ операција.



Слика 4. Рута $x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$ *се операцијом* $swapTwo(2)$ *трансформише у руту* $x' = [0, 1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11]$.

5.3.2 $Swap$

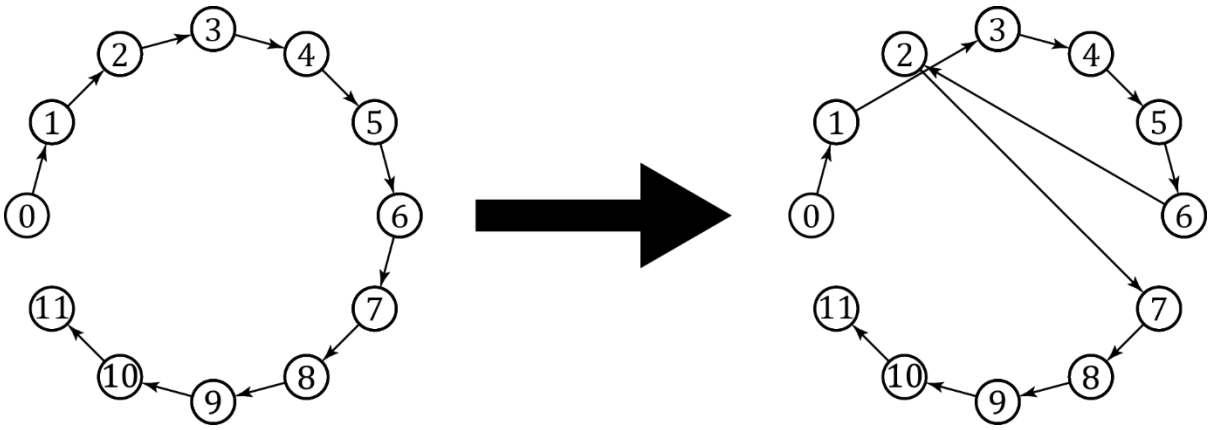
$Swap$ операција узима као аргумент два броја и замењује места чворовима под тим индексима у тренутној рути. Ово је уопштење операције $SwapTwo$ и сложеност претраге околине дефинисане овом операцијом је $O(n^2)$.



Слика 5. Рута $x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$ *се операцијом* $swap(2, 6)$ *трансформише у руту* $x' = [0, 1, 6, 3, 4, 5, 2, 7, 8, 9, 10, 11]$.

5.3.3 $RemoveInsert$

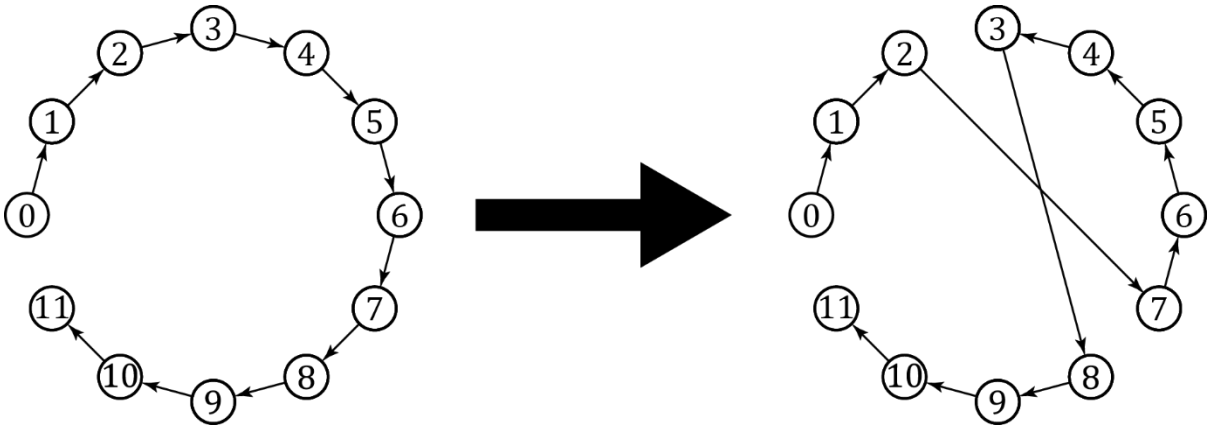
Ова операција такође зависи од два аргумента, индекс места у рути са којег треба уклонити чвор i и индекс места где тај чвор треба сместити, j. За испитивање целе $removeInsert$ околине, треба извршити $O(n^3)$ операција због промене позиције чворова на позицијама између i и j.



Слика 6. Рута $x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$ *се операцијом* $removeInsert(2, 6)$ *трансформише у руту* $x' = [0, 1, 3, 4, 5, 6, 2, 7, 8, 9, 10, 11]$.

5.3.4 $2\text{-}opt$

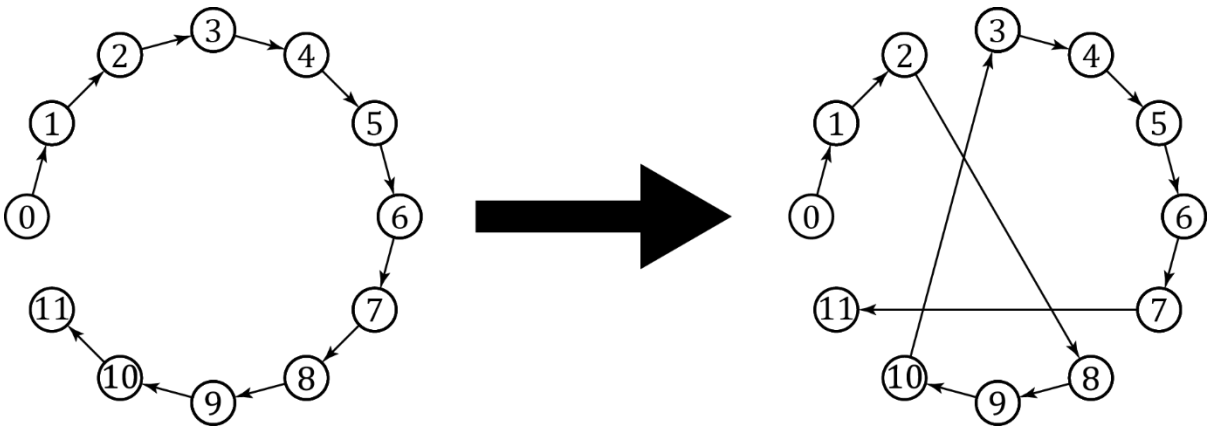
$2\text{-}opt$ је нешто сложенија операција која се састоји из уклањања две гране из датог пута и додавања две нове гране. Овој операцији треба проследити два параметра, тј. индексе, полазних чворова грана које се уклањају из тренутне руте. Начин додавања две нове гране приказан је на слици 7. $2\text{-}opt$ захтева промену смера обиласка дела пута између две гране које се уклањају па је сложеност претраге целе околине $O(n^3)$.



Слика 7. Рута $x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$ *се операцијом* $2\text{-}opt(2, 7)$ *трансформише у руту* $x' = [0, 1, 2, 7, 6, 5, 4, 3, 8, 9, 10, 11]$.

5.3.5 $Or\text{-}opt$

Операција $or\text{-}opt$ уклања три гране из тренутне руте и додаје три нове гране тако да је резултујући пут допустив. Параметри који се прослеђују су полазни чворови три гране, редом i, j и k. Сложеност претраге ове околине је $O(n^4)$ због премештања чворова између грана које се уклањају.



Слика 8. Рута $x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$ *се операцијом* $or\text{-}opt(2, 7, 10)$ *трансформише у руту* $x' = [0, 1, 2, 8, 9, 10, 3, 4, 5, 6, 7, 11]$.

5.4 Динамичко рачунање функције циља

За проблем трговачког путника, израчунавање функције циља после промене у некој околини захтева константан број операција. Ово није случај код MLP-а где се промена једног чвора одражава на све чворове који се налазе након њега у рути. Стога, са директним рачунањем функције циља, сложеност претраге околине расте за мултипликативни фактор $O(n)$ (на пример, за претрагу $swap$ околине, са циљем проналасна бољег решења, потребно је $O(n^3)$ операција). Овакво директно рачунање захтева промену решења како би било могуће. Да би се избегло директно израчунавање функције циља, у [15] је предложена провера потеза у околини (тј. провера да ли је ново решење боље од старог) која има амортизовану временску сложеност $O(1)$ и које нема потребе за променом решења како би се израчунала функција циља. У овом раду се предлаже сличан механизам провере квалитета новог решења, заснован на [16].

Механизам испитивања новог решења састоји се из конкатенације сегмената на које се рута раставља приликом потеза у околини, при чему се користе глобалне особине тренутног пута и тих сегмената. Потребно је дефинисати ове глобалне податке како би се могао испитати сваки нови потез у некој околини.

Нека је $\sigma = (\sigma_1, \dots, \sigma_k)$ сегмент руте. Тада се уводе:

- $T(\sigma)$ – представља време потребно за обилазак сегмента σ
- $C(\sigma)$ – представља цену обиласка сегмента σ у смислу збира кашњења
- $R(\sigma)$ – представља број чворова у сегменту σ

За семент σ који се састоји од једног чвора важи $T(\sigma) = 0$, цена обиласка $C(\sigma) = 0$ и $R(\sigma) = 0$ ако је чвор 0-ти, иначе $R(\sigma) = 1$.

Нека је $\sigma = (\sigma_1, \dots, \sigma_k)$ и $\sigma' = (\sigma'_1, \dots, \sigma'_k)$. За конкатенацију два сегмента важе правила:

$$T(\sigma \oplus \sigma') = T(\sigma) + c_{[\sigma_k][\sigma'_1]} + T(\sigma') \tag{5.1}$$

$$C(\sigma \oplus \sigma') = C(\sigma) + \mathrm{R}(\sigma')\left(T(\sigma) + c_{[\sigma_k][\sigma'_1]}\right) + C(\sigma') \tag{5.2}$$

$$R(\sigma \oplus \sigma') = R(\sigma) + \mathrm{R}(\sigma') \tag{5.3}$$

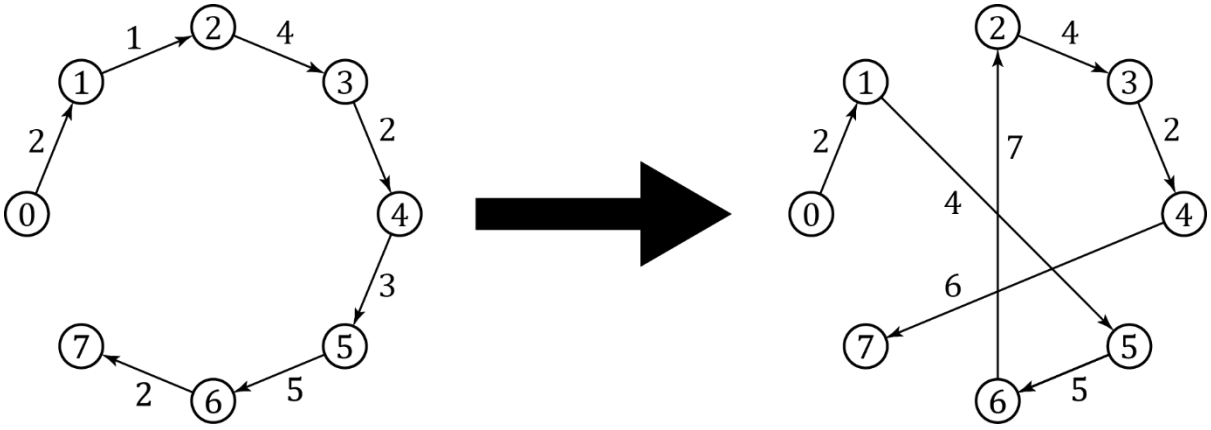
где је $\oplus$ оператор конкатенације а $c_{[\sigma_k][\sigma'_1]}$ трошак пута од чвора на позицији σ_k у рути до чвора на позицији σ'_1. Ако је сегмент σ' сегмент σ у обрнутом смеру, тада $T(\sigma) = T(\sigma')$, $R(\sigma) = \mathrm{R}(\sigma')$ а $C(\sigma') = kT(\sigma) - C(\sigma)$, што омогућава лаку конкатенацију сегмента који промене смер што се користи у $2\text{-}opt$ околини.

Како би било могуће овакво динамичко испитивање новог решења, потребно је извршити предпроцесирање уведених глобалних података за сваки сегмент тренутног пута. Ова операција има сложеност $O(n^2)$ и извршава се након сваке промене решења.

Примећује се такође да тада нема потребе за променом решења како би се одредила његова функција циља (а тиме одредило да ли је боље од тренутног решења), већ се та вредност може израчунати користећи правила (5.1)-(5.3) над одговарајућим сегментима тренутне околине у константном времену. Ово смањује сложеност претраге околина $removeInsert$ и $2\text{-}opt$ на $O(n^2)$ а $or\text{-}opt$ околине на $O(n^3)$ што значајно умањује време извршавања хибридног алгоритма.

5.4.1 Пример

На слици 9 је дата инстанца са 8 чворова, њено почетно решење и изглед новог решења које се добија операцијом $or\text{-}opt$, заједно са временима потребним за обилазак грана на путу.



Слика 9. Рута $x = [0, 1, 2, 3, 4, 5, 6, 7]$ и решење у $or\text{-}opt$ околини x.

Да би се искористила динамичка провера цене новог решења са слике 9, потребно је спојити сегменте $(0, 1)$, $(5, 6)$, $(2, 3, 4)$ и (7), а потребно је и познавање већ поменутих глобалних информација о овим сегментима. Сада се могу извршити конкатенације и може се проверити цена без претходне промене почетног решења. Овај процес који користи правила (5.1)-(5.3), као и глобални подаци, дати су у следећој табели.

Табела 1. Конкатенације сегмената

σ	$T(\sigma)$	$C(\sigma)$	$R(\sigma)$
$(0, 1)$	2	2	1
$(5, 6)$	5	5	2
$(2, 3, 4)$	6	10	3
(7)	0	0	1
$(0, 1) \oplus (5, 6)$	11	19	3
$(0, 1) \oplus (5, 6) \oplus (2,3,4)$	24	83	6
$(0, 1) \oplus (5, 6) \oplus (2,3,4) \oplus (7)$	30	**113**	7

Када се израчуна цена новог решења, може се закључити да ли је боље од старог и према томе извршити ажурирање решења.

5.5 Шема хлађења

У имплементацији хибридног алгоритма, температура T се на почетку симулираног каљења иницијализује на вредност $T_{MAX} = \lfloor n * 6 \rfloor$ где је n број чворова инстанце која се решава. Параметар T се мења у свакој итерацији геометријском прогресијом са фактором $\gamma = 0{,}9$ док не достигне минималну температуру $T_{MIN} = \lfloor n/2 \rfloor$. Број итерација на свакој температури износи $M_{MAX} = \frac{\lfloor n*6 \rfloor}{T}$ па број итерација на почетку износи 1 а при крају извршавања алгоритма SA око 12. Ови параметри су добијени емиријски.

6 Експериментална анализа

Хибридни алгоритам изложен у овом раду, имплементиран је у програмском језику *C++* користећи *Visual Studio 2013* окружење на *Windows 8.1 Pro 64bit* оперативном систему. Тест окружење је било *Intel i7-3770k @4.0Ghz* са *16GB RAM*-а, а при извршавању коришћен је само један *thread*.

6.1 Инстанце

Алгоритам је тестиран на два скупа инстанци. како и на скупу инстанци генерисаних на случајан начин. Први скуп, одабран у [9], је димензија инстанци од 42 до 107 чворова и припада јавно доступној библиотеци *TSPLIB* за проблем трговачког путника. Други скуп представља инстанце генерисане на случајан начин и представљене у [13] где су тестиране. У овом раду из другог скупа тестираће се инстанце димензија $10, 20, 50, 100$ и 200 где су координате чворова из униформне расподеле $U[1, 100]$ и инстанце димензије 500 чије су координате чворова из $U[1, 500]$. Свака група различите димензије има по 20 инстанци.

6.2 Резултати

Свака инстанца тестирана је по 10 пута како би се добио увид у просечно решење (колона *пр. реш.* у табелама) које постиже хибридни алгоритам као и у средње одступање (у табелама, колона *ср. одст.*) од оптималног решења (горњег ограничења). Оптимално решење или горње ограничење се у табелама налази у колони *опт. реш.* Просечно одступање у процентима се рачуна као *ср. одст.* $= 100\frac{(\text{пр.реш.} - \text{опт.реш.})}{\text{опт.реш.}}$. Време у табелама (колона *време*) представља просечно време од 10 извршавања алгоритма на истој инстанци проблема.

У табели 2, приказан је скуп инстанци из библиотеке $TSPLIB$, одабран у [9] где су добијена оптимална решења за све осим две инстанце. Касније је у [16] изложен алгоритам $GILS\text{-}RVND$ који је за те две инстанце поправио горње ограничење а на

осталим достигао оптимална па се резултати тестирања упоређују са тим алгоритмом. Треба напоменути да су решења у овој табели Хамилтонови циклуси а не путеви.

Оптималне вредности у табелама су зацрњене.

Табела 2. Резултати тестирања инстанци одабраних у [9].

	GILS-RVND		*VNS + SA*			
инст.	*опт. реш.*	*време (s)*	*мин. реш.*	*пр. реш.*	*ср. одст. (%)*	*време (s)*
dantzig42	**12.528**	0,16	**12.528**	12.528	0,00	0,21
swiss42	**22.327**	0,16	**22.327**	22.327	0,00	0,20
att48	**209.320**	0,32	**209.320**	209.320	0,00	0,29
gr48	**102.378**	0,33	**102.378**	102.378	0,00	0,27
hk48	**247.926**	0,30	**247.926**	248.069	0,06	0,26
eil51	**10.178**	0,49	**10.178**	10.249	0,69	0,29
berlin52	**143.721**	0,46	**143.721**	143.863	0,10	0,33
brazil58	**512.361**	0,78	**512.361**	512.361	0,00	0,46
st70	**20.557**	1,65	**20.557**	20.557	0,00	0,70
eil76	**17.976**	2,64	**17.976**	18.070	0,52	0,87
pr76	**3.455.242**	2,31	**3.455.242**	3.459.378	0,12	0,94
gr96	**2.097.170**	6,19	**2.097.170**	2.101.268	0,20	1,68
rat99	57.986*	11,27	57.986	58.211	0,39	2,15
kroA100	**983.128**	8,59	**983.128**	992.584	0,96	2,30
kroB100	**986.008**	9,21	**986.008**	986.008	0,00	2,16
kroC100	**961.324**	8,17	**961.324**	961.324	0,00	2,12
kroD100	**976.965**	8,46	**976.965**	977.748	0,08	2,10
kroE100	**971.266**	8,31	**971.266**	971.354	0,01	2,03
rd100	**340.047**	8,52	**340.047**	340.579	0,16	1,95
eil101	27.519*	12,76	27.519	27.717	0,74	2,10
lin105	**603.910**	8,42	**603.910**	603.910	0,00	2,36
pr107	**2.026.626**	10,89	**2.026.626**	2.026.626	0,00	2,66

* *оптималност решења није доказана, вредност представља горње ограничење*

Може се приметити да је алгоритам изложен у овом раду за све инстанце одабране у [9] достигао оптимална решења или најбоље познато горње ограничење за највише пар секунди уз највеће средње одступање од $0{,}96\%$.

Табела 3. Резултат тестирања инстанци димензије 10 конструисаних у [13].

		VNS + SA			
инст.	*опт. реш.*	*мин. реш.*	*пр. реш.*	*ср. одст. (%)*	*време (s)*
TRP-S10-R1	**1.303**	**1.303**	1.303	0,00	0,04
TRP-S10-R2	**1.517**	**1.517**	1.517	0,00	0,04
TRP-S10-R3	**1.233**	**1.233**	1.233	0,00	0,04
TRP-S10-R4	**1.386**	**1.386**	1.386	0,00	0,04
TRP-S10-R5	**978**	**978**	978	0,00	0,04
TRP-S10-R6	**1.477**	**1.477**	1.477	0,00	0,04
TRP-S10-R7	**1.163**	**1.163**	1.163	0,00	0,04
TRP-S10-R8	**1.234**	**1.234**	1.234	0,00	0,04
TRP-S10-R9	**1.402**	**1.402**	1.402	0,00	0,04
TRP-S10-R10	**1.388**	**1.388**	1.388	0,00	0,04
TRP-S10-R11	**1.405**	**1.405**	1.405	0,00	0,04
TRP-S10-R12	**1.150**	**1.150**	1.150	0,00	0,04
TRP-S10-R13	**1.531**	**1.531**	1.531	0,00	0,04
TRP-S10-R14	**1.219**	**1.219**	1.219	0,00	0,04
TRP-S10-R15	**1.087**	**1.087**	1.087	0,00	0,04
TRP-S10-R16	**1.264**	**1.264**	1.264	0,00	0,04
TRP-S10-R17	**1.058**	**1.058**	1.058	0,00	0,04
TRP-S10-R18	**1.083**	**1.083**	1.083	0,00	0,04
TRP-S10-R19	**1.394**	**1.394**	1.394	0,00	0,04
TRP-S10-R20	**951**	**951**	951	0,00	0,04

У табели 3 су тестиране инстанце димензије 10 конструисане у [13] на случајан начин су решене оптимално са средњим одступањем једнаким 0. За инстанце димензија 30 и 50 се такође добија оптимално решење, што се види из табела 4 и 5. Средње одступање за инстанце димензије 30 само у једном случају има вредност

различиту од 0, а код димензије 50 у 8 случајева је већа од 0 са највећом вредношћу $0{,}45\%$.

Табела 4. Резултат тестирања инстанци димензије 30 конструисаних у [13].

		VNS + SA			
инст.	*опт. реш.*	*мин. реш.*	*пр. реш.*	*ср. одст. (%)*	*време (s)*
TRP-S30-R1	**3.175**	**3.175**	3.175	0,00	0,07
TRP-S30-R2	**3.248**	**3.248**	3.249	0,02	0,05
TRP-S30-R3	**3.570**	**3.570**	3.570	0,00	0,07
TRP-S30-R4	**2.983**	**2.983**	2.983	0,00	0,07
TRP-S30-R5	**3.248**	**3.248**	3.248	0,00	0,06
TRP-S30-R6	**3.328**	**3.328**	3.328	0,00	0,06
TRP-S30-R7	**2.809**	**2.809**	2.809	0,00	0,06
TRP-S30-R8	**3.461**	**3.461**	3.461	0,00	0,06
TRP-S30-R9	**3.475**	**3.475**	3.475	0,00	0,07
TRP-S30-R10	**3.359**	**3.359**	3.359	0,00	0,06
TRP-S30-R11	**2.916**	**2.916**	2.916	0,00	0,06
TRP-S30-R12	**3.314**	**3.314**	3.314	0,00	0,07
TRP-S30-R13	**3.412**	**3.412**	3.412	0,00	0,07
TRP-S30-R14	**3.297**	**3.297**	3.297	0,00	0,06
TRP-S30-R15	**2.862**	**2.862**	2.862	0,00	0,06
TRP-S30-R16	**3.433**	**3.433**	3.433	0,00	0,06
TRP-S30-R17	**2.913**	**2.913**	2.913	0,00	0,06
TRP-S30-R18	**3.124**	**3.124**	3.124	0,00	0,06
TRP-S30-R19	**3.299**	**3.299**	3.299	0,00	0,07
TRP-S30-R20	**2.796**	**2.796**	2.796	0,00	0,07

У табели 6, приказани су резултати тестирања инстанци димензије 100 конструисаних у [13], где је познато само горње ограничење које је касније побољшано у [16]. Стога се резултати упоређују са резултатима из [16]. За 19 инстанци је достигнуто најбоље познато решење а средње одсупање на свим инстанцама не прелази $0{,}75\%$. Просечно време извршавања је највише $2{,}19$ секунди.

У табели 7, приказани су резултати тестирања инстанци димензије 200, које су конструисане у [13]. Најбоља горња ограничења су добијена у [16] па се резултати упоређују са резултатима из тог рада. За 9 инстанци се достиже најбоље познато горње ограничење уз средње одступање на свим инстанцама највише 1,74%. Највеће просечно време извршавања је 21,49 секунди.

У табели 8, упоређују се резултати на инстанцама димензије 500 генерисаних у [13] где су најбоља горња ограничења добијена у [16]. Средње оступање је највише 2,79% а просечно време извршавања око 500 секунди.

Табела 5. Резултат тестирања инстанци димензије 50 конструисаних у [13].

		VNS + SA			
инст.	*опт. реш.*	*мин. реш.*	*пр. реш.*	*ср. одст. (%)*	*време (s)*
TRP-S50-R1	**12.198**	**12.198**	12.198	0,00	0,30
TRP-S50-R2	**11.621**	**11.621**	11.621	0,00	0,30
TRP-S50-R3	**12.139**	**12.139**	12.139	0,00	0,30
TRP-S50-R4	**13.071**	**13.071**	13.163	0,70	0,30
TRP-S50-R5	**12.126**	**12.126**	12.126	0,00	0,29
TRP-S50-R6	**12.684**	**12.684**	12.685	0,01	0,30
TRP-S50-R7	**11.176**	**11.176**	11.176	0,00	0,31
TRP-S50-R8	**12.910**	**12.910**	12.912	0,01	0,30
TRP-S50-R9	**13.149**	**13.149**	13.149	0,00	0,29
TRP-S50-R10	**12.892**	**12.892**	12.892	0,00	0,32
TRP-S50-R11	**12.103**	**12.103**	12.106	0,03	0,30
TRP-S50-R12	**10.633**	**10.633**	10.633	0,00	0,31
TRP-S50-R13	**12.115**	**12.115**	12.115	0,00	0,30
TRP-S50-R14	**13.117**	**13.117**	13.129	0,09	0,29
TRP-S50-R15	**11.986**	**11.986**	11.986	0,00	0,32
TRP-S50-R16	**12.138**	**12.138**	12.138	0,00	0,31
TRP-S50-R17	**12.176**	**12.176**	12.205	0,24	0,33
TRP-S50-R18	**13.357**	**13.357**	13.420	0,47	0,31
TRP-S50-R19	**11.430**	**11.430**	11.430	0,00	0,33
TRP-S50-R20	**11.935**	**11.935**	11.938	0,02	0,30

У табели 9 су приказани резултати тестирања скупа инстанци из библиотеке $TSPLIB$ које су одабране у [13]. Најбоља горња ограничења су такође добијена у [16] па се резултати упоређују са тим ограничењима.

Табела 6. Резултат тестирања инстанци димензије 100 конструисаних у [13].

	GILS-RVND		*VNS + SA*			
инст.	*мин. реш.*	*време (s)*	*мин. реш.*	*пр. реш.*	*ср. одст. (%)*	*време (s)*
TRP-S100-R1	32.779	7,05	32.779	32.788	0,03	1,86
TRP-S100-R2	33.435	7,51	33.435	33.457	0,07	1,96
TRP-S100-R3	32.390	7,07	32.390	32.446	0,17	1,98
TRP-S100-R4	34.733	7,27	34.733	34.843	0,32	2,06
TRP-S100-R5	32.598	8,87	32.598	32.638	0,12	1,99
TRP-S100-R6	34.159	7,82	34.159	34.336	0,52	1,98
TRP-S100-R7	33.375	8,74	33.375	33.450	0,23	2,09
TRP-S100-R8	31.780	7,08	31.780	31.829	0,15	2,00
TRP-S100-R9	34.167	7,47	34.167	34.258	0,27	1,93
TRP-S100-R10	31.605	6,78	31.605	31.605	0,00	1,97
TRP-S100-R11	34.188	7,75	34.188	34.419	0,68	2,02
TRP-S100-R12	32.146	7,20	32.146	32.260	0,36	2,13
TRP-S100-R13	32.604	7,78	32.604	32.848	0,75	2,11
TRP-S100-R14	32.433	6,29	32.438	32.438	0,02	2,15
TRP-S100-R15	32.574	7,13	32.574	32.638	0,19	1,94
TRP-S100-R16	33.566	7,97	33.566	33.568	0,01	2,17
TRP-S100-R17	34.198	9,23	34.198	34.234	0,11	2,19
TRP-S100-R18	31.929	7,07	31.929	32.077	0,46	2,07
TRP-S100-R19	33.463	8,08	33.463	33.529	0,20	1,97
TRP-S100-R20	33.632	8,73	33.632	33.732	0,30	1,99

Табела 7. Резултат тестирања инстанци димензије 200 конструисаних у [13].

	GILS-RVND		*VNS + SA*			
инст.	*мин. реш.*	*време (s)*	*мин. реш.*	*пр. реш.*	*ср. одст. (%)*	*време (s)*
TRP-S200-R1	88.787	73,39	88.806	89.272	0,55	20,20
TRP-S200-R2	91.977	68,07	92.448	93.116	1,24	19,06
TRP-S200-R3	92.568	67,11	92.568	93.204	0,69	20,00
TRP-S200-R4	93.174	72,17	93.174	93.827	0,70	19,46
TRP-S200-R5	88.737	70,77	88.737	89.766	1,16	18,06
TRP-S200-R6	91.589	70,52	91.589	92.814	1,34	19,78
TRP-S200-R7	92.754	72,80	92.756	93.846	1,18	21,49
TRP-S200-R8	89.048	75,15	89.118	89.786	0,83	19,12
TRP-S200-R9	86.326	65,45	86.326	86.945	0,72	19,32
TRP-S200-R10	91.552	74,25	91.650	92.392	0,92	18,93
TRP-S200-R11	92.655	73,15	92.815	93.301	0,70	19,76
TRP-S200-R12	91.457	76,74	91.457	92.060	0,66	19,12
TRP-S200-R13	86.155	72,96	86.155	86.909	0,88	18,64
TRP-S200-R14	91.882	70,94	91.989	92.607	0,79	20,04
TRP-S200-R15	88.912	70,41	91.589	89.737	0,93	19,39
TRP-S200-R16	89.311	77,89	89.311	90.866	1,74	20,72
TRP-S200-R17	89.089	71,17	89.089	89.777	0,77	19,79
TRP-S200-R18	93.619	77,03	93.972	94.697	1,15	19,53
TRP-S200-R19	93.369	71,08	93.676	94.574	1,29	18,51
TRP-S200-R20	86.292	70,61	86.331	87.347	1,22	20,14

Табела 8. Резултат тестирања инстанци димензије 500 конструисаних у [13].

	GILS-RVND		*VNS + SA*			
инст.	*мин. реш.*	*време (s)*	*мин. реш.*	*пр. реш.*	*ср. одст. (%)*	*време (s)*
TRP-S500-R1	1.841.386	1738,48	1.871.090	1.889.008	2,59	483,38
TRP-S500-R2	1.816.568	1476,13	1.835.543	1.847.732	1,72	525,37
TRP-S500-R3	1.833.044	1557,48	1.845.870	1.866.666	1,83	513,50
TRP-S500-R4	1.809.266	1597,06	1.821.669	1.846.250	2,04	489,95
TRP-S500-R5	1.823.975	1530,94	1.842.106	1.864.735	2,23	463,15
TRP-S500-R6	1.786.620	1576,91	1.812.489	1.826.810	2,25	510,18
TRP-S500-R7	1.847.999	1584,67	1.875.923	1.901.250	2,88	458,71
TRP-S500-R8	1.820.846	1565,01	1.844.134	1.864.047	2,37	477,74
TRP-S500-R9	1.733.819	1409,23	1.751.823	1.764.179	1,75	478,03
TRP-S500-R10	1.762.741	1621,85	1.797.515	1.804.639	2,38	455,73
TRP-S500-R11	1.797.881	1530,98	1.810.796	1.830.966	1,84	471,39
TRP-S500-R12	1.774.452	1554,75	1.795.137	1.820.156	2,58	489,39
TRP-S500-R13	1.873.699	1598,46	1.900.017	1.909.469	1,91	457,19
TRP-S500-R14	1.799.171	1701,90	1.822.190	1.829.656	1,69	460,45
TRP-S500-R15	1.791.145	1623,79	1.801.765	1.823.544	1,81	476,89
TRP-S500-R16	1.810.188	1583,70	1.834.573	1.840.784	1,69	451,12
TRP-S500-R17	1.825.748	1549,80	1.857.894	1.871.635	2,51	488,32
TRP-S500-R18	1.826.263	1620,02	1.864.287	1.876.765	2,77	497,81
TRP-S500-R19	1.779.248	1602,87	1.825.407	1.828.812	2,79	501,49
TRP-S500-R20	1.820.813	1507,96	1.847.916	1.849.547	1,58	455,57

Табела 9. Скуп инстанци из библиотеке $TSPLIB$ одабраних у [13].

	GILS-RVND		*VNS + SA*			
инст.	*мин. реш.*	*време (s)*	*мин. реш.*	*пр. реш.*	*ср. одст. (%)*	*време (s)*
st70	19.215	1,51	19.710	19.751	2,79	0,59
rat99	54.984	9,47	56.573	56.862	3,42	1,77
kroD100	949.594	6,90	951.609	951.944	0,25	1,70
lin105	585.823	6,19	586.751	586.751	0,16	1,75
pr107	1.980.767	8,13	1.981.991	1.981.995	0,06	2,13
rat195	210.191	75,56	217.104	218.160	3,79	13,54
pr226	7.100.308	59,05	7.118.051	7.118.796	0,26	25,77
lin318	5.560.679	220,59	5.569.520	5.613.412	0,95	79,35
pr439	17.688.561	553,74	17.800.061	17.855.062	0,94	246,79
att532	5.581.240	1792,61	5.652.178	5.691.485	1,98	601,87

7 Закључак

У овом раду описане су две метахеуристике које имају широку примену у приближном решавању разних оптимизационих проблема, који се за велике димензије не могу решити практично тачним решавачем. Описане метахеуристике су метода променљивих околина и метода симулираног каљења, за које је изложен псеудокод и значајније особине алгоритама.

Изложен је проблем минималног кашњења, описан његов значај у пракси и дата математичка формулација проблема. Развијена је хибридна метахеуристика која спаја методу промељивих околина и методу симулираног каљења. У фази локалне претраге овог хибридног алгоритма се на почетку користи метода VND како би се добило добро међурешење које се у другом делу алгоритма унапређује методом симулираног каљења. Дат је детаљан опис имплементације, објашњене су структуре коришћених околина. Описана је динамичка провера квалитета решења која се користи приликом претраге околине. Ова динамичка провера значајно смањује сложеност алгоритма а тиме и време извршавања.

Хибридни алгоритам методе променљивих околина и методе симулираног каљења за проблем минималног кашњења је први пут описан у овом раду. Имплементација овог алгоритма је тестирана на јавно доступним инстанцама библиотеке $TSPLIB$ као и на скупу инстанци конструисаних на случајан начин у [13]. Постигнута су тренутно најбоља решења из [16] за $TSPLIB$ инстанце димензија 42 до 107 одабране у [9], уз највеће средње одступање од $0{,}96\%$ и времена од $0{,}21$ до $2{,}66$ секунди. За инстанце генерисане у [13] постигута су оптимална решења за све инстанце димензија $10, 30$ и 50, а за инстанце димензије 100, за 19 од 20 инстанци је достигнуто горње ограничење из [16] где су средња одступања за све инстанце мање од $0{,}75\%$. За инстанце из [13] димензија 200 и 500 постигнута су решења близу тренутно најбољим из [16] са средњим одступањем мањим од 2.88% уз знатно краће време извршавања. На последњем скупу инстанци које припадају библиотеци $TSPLIB$, одабране у [13] а чија су најбоља решења постигнута у [16], хибридни алгоритам описан у овом раду достиже решења близу тренутно најбољим са средњим одступањем највише $3{,}79\%$ и битно краћим временом извршавања. За 102 од 109 инстанци димензија 42 до 107 постигнута су оптимална решења или најбоља горња ограничења која су тренутно позната. На инстанцама димензија 195 до 532 постигута су решења близу најбољих горњих ограничења.

Како нису достигнута најбоља позната решења за инстанце димензија већих од 195, постоји простор за даље усавршавање овог хибридног алгоритма. Даљи рад на алгоритму обухвата додавање нових структура околина како би се утврдила најбоља комбинација постојећих и нових околина, као и разматрање других хеуристичких метода које би се примењивале у фази локалне претраге и њихова најбоља комбинација.

Литература

1. **Hansen, P., Mladenovic, N., Moreno Perez, J.A.** Variable neighborhood search: methods and applications. *4OR-Q J Oper. Res.* 2008, T. 6, str. 319-360.

2. **Mladenovic N., Hansen E.** Variable neighborhood search. *Computers & Operations Research.* 1997, T. 24, 11, str. 1097-1100.

3. **Glover F., Kochenberger G.A.** *Handbook of Metaheuristics.* [ur.] Potvin J.-Y. Gendreau M. 2nd. s.l. : Springer, 2010. T. 146.

4. **Francisco Angel-Bello Acosta, Ada Alvarez Socarras, Irma Garcia.** *Formulation for the minimum latency problem: an experimental evaluation.* 2011.

5. **Sahni, S., Gonzalez, T.** P-complete approximation problems. *Journal of the AXM.* 23, 1976, T. 3, str. 555-565.

6. **Lucena, A.** Time-dependent traveling salesman problem - the deliveryman case. *Networks.* 20, 1990, str. 753-763.

7. **Fischetti M., Laporte G., Martello S.** The delivery man problem and cumulative matroids. *Operations Research.* 41, 1993, T. 6, 1055-1064.

8. **Mendez-Diaz P., Zabala P.,Lucena A.** A new formulation for the traveling deliveryman problem. *Discrete Appled Math.* 156, 2008, str. 3223-3237.

9. **Abelado H., Fukasawa R., Pessoa A. Uchoa E.** The time dependent traveling salesman problem: Polyhedra and algorithm. *Tech. Rep. RPEP.* 10, 2010, T. 15.

10. **Abeledo H., Fukasawa R., Pessoa A., Uchoa E.** The time dependent traveling salesman problem: Polyhedra and branch-cut-and-price algorithm. *Proceedings of the 9th International Symposium on Experimental Algorithms.* 2010, str. 202-2013.

11. **Blum A., Chalasanit P., Pulleyblankt B., Raghavan P., Sudan M.** The minimum latency problem. *Proceedings of the 26th Annual ACM Symposium on Theory of Computing.* 1994, str. 163–171.

12. **Chaudhuri K., Godfrey B., Rao S., Talwar K.** Paths, trees, and minimum latency tours. *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science.* 2003, str. 36-45.

13. **Salehipour, A., Sörensen, K., Goos, P., Bräysy, O.** Efficient GRASP + VND and GRASP + VNS metaheuristics for the traveling repairman problem. *A Quarterly Journal of Operations Research.* 9, 2011, T. 2, str. 189-209.

14. **Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F.C.R., Vansteenwegen, P.** Heuristics for the traveling repairman problem with profits. *Proceedings of the 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2010.* 2010, str. 34-44.

15. **Ngueveu, S., Prins, C., Wolfler Calvo, R.** An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research.* 2010, T. 37, 11, str. 1877–1885.

16. **Melo Silva M., Subramanian A., Vidal T., Satoru Ochi L.** A simple and effective metaheuristic for the Minimum Latency Problem. *European Journal of Operational Research.* 2012, T. 221, str. 513-520.

17. **Blum C., Jose Blesa Aguilera M., Roli A., Sampels M.** Hybrid Metaheuristics: An Emerging Approach to Optimization. s.l. : Springer, 2008.