УНИВЕРЗИТЕТ У БЕОГРАДУ

МАТЕМАТИЧКИ ФАКУЛТЕТ

Стефан В. Капунац

Методе за ефикасно решавање доминацијских проблема на великим графовима

докторска дисертација

Београд, 2026.

UNIVERSITY OF BELGRADE

FACULTY OF MATHEMATICS

Stefan V. Kapunac

Methods for efficient solving of domination problems on large graphs

doctoral dissertation

Belgrade, 2026.

Ментор:
др Александар Картељ, ванредни професор
Универзитет у Београду, Математички факултет

Чланови комисије:
др Владимир Филиповић, редовни професор
Универзитет у Београду, Математички факултет

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

др Марко Ђукановић, доцент
Универзитет у Бањој Луци, Природно-математички факултет

Датум одбране:

Наслов дисертације: Методе за ефикасно решавање доминацијских проблема на великим графовима

Сажетак: У овој дисертацији разматрају се методе за ефикасно решавање више важних варијанти доминацијских проблема на графовима, са посебним осврtом на инстанце великих димензија које се често јављају у реалним системима. Доминацијски проблеми имају бројне примене у анализи и управљању комплексним мрежама, укључујући друштвене, телекомуникационе, транспортне и биолошке мреже. Истраживање обухвата четири проблема: минималну тежинску тоталну доминацију, минималну тежинску независну доминацију, k-јаку римску доминацију и основни проблем минималне доминације на великим графовима.

За проблем минималне тежинске тоталне доминације предложен је метахеуристички приступ заснован на методи променљивих околина, са пажљиво дефинисаним механизмима размрдавања, локалне претраге и функције прилагођености. Резултати показују да предложени алгоритам постиже оптимална решења на малим и средњим инстанцама и надмашује конкурентске приступе на великим графовима. Додатно, предложена је примена овог проблема за убрзавање ширења информација на друштвеним мрежама.

За проблем минималне тежинске независне доминације развијена су два нова модела целобројног линеарног програмирања. Решавањем ових модела проналазе се оптимална решења на свим мањим инстанцама, а показују супериорније перформансе у односу на конкурентске егзактне приступе и на већим графовима. Поред тога, предложена је похлепна хеуристика која надмашује конкурентске похлепне приступе на већини инстанци.

У случају k-јаке римске доминације, развијена је похлепна хеуристика заснована на информацијама о покривености чворова, као и метахеуристички приступ заснован на методи променљивих околина који користи поменути похлепни алгоритам за иницијализацију. Овај проблем је изузетно изазован због експоненцијалне сложености провере допустивости решења, те је уведен концепт квазидопустивости који омогућава ефикасну процену допустивости током претраге. Експериментални резултати показују да предложени алгоритам доследно надмашује похлепни приступ и постојеће конкурентске методе, поготово на већим графовима. Практична вредност алгоритма илустрована је кроз студију случаја која обухвата оптимално позиционирање ватрогасних станица и возила у градским општинама, како би читав град био безбедан у условима појаве k истовремених пожара.

За проблем минималне доминације предложен је нови хибридни приступ IRIS. IRIS је дизајниран као оквир опште намене који превазилази јаз између егзактних решавача целобројног линеарног програмирања и хеуристичке претраге, итеративним фиксирањем одабраних променљивих ради смањења простора претраге. Научни допринос огледа се у флексибилном механизму конструкције потпроблема, који се може прилагодити коришћењем различитих стратегија селекције. У овом раду имплементирана је и евалуирана специфична конфигурација IRIS-а која користи историјске статистичке податке и хеуристику засновану на покривености чворова за интелигентно идентификовање променљивих које ће бити фиксиране. Овакав циљани приступ омогућава ILP решавачу да пронађе квалитетна решења за инстанце великих димензија које су рачунски неизводљиве за егзактне методе. Експериментални резултати показују да IRIS остварује конкурентне перформансе у односу на најбоље постојеће методе, што га чини валидном алтернативом за решавање доминацијских и потенцијално других НП-тешких проблема.

Кључне речи: доминацијски проблеми на графовима, метода променљивих околина, целобројно линеарно програмирање, хибридни алгоритми, велики графови, комплексне мреже

Научна област: Рачунарство и информатика
Ужа научна област: Оптимизација

Dissertation title: Methods for efficient solving of domination problems on large graphs

Abstract: This dissertation addresses methods for efficiently solving several important variants of domination problems on graphs, with a particular focus on large-scale instances that frequently appear in real-world systems. Domination problems have numerous applications in the analysis and management of complex networks, including social, telecommunication, transport, and biological networks. The study covers four problems: minimum weight total domination, minimum weight independent domination, k-strong Roman domination, and the canonical minimum domination problem on large graphs.

For the minimum weight total domination problem, a variable neighborhood search approach is proposed, with carefully designed mechanisms for shaking, local search, and fitness function evaluation. The results show that the proposed algorithm achieves optimal solutions on small and medium instances and outperforms competing approaches on large graphs. Additionally, an application of this problem for accelerating information spreading in social networks is proposed.

For the minimum weight independent domination problem, two new integer linear programming models are developed. Solving these models finds optimal solutions for all smaller instances while demonstrating superior performance compared to competing exact approaches on larger graphs. In addition, a greedy heuristic is proposed that outperforms competing greedy methods on most instances.

In the case of k-strong Roman domination, a greedy heuristic based on node coverage information is developed, along with a metaheuristic approach based on variable neighborhood search that uses the greedy algorithm for initialization. This problem is particularly challenging due to the exponential complexity of solution feasibility verification, leading to the introduction of the concept of quasi-feasibility that enables efficient feasibility assessment during the search. Experimental results show that the proposed algorithm consistently outperforms the greedy approach and existing competing methods, especially on larger graphs. The practical value of the algorithm is illustrated through a case study involving the optimal positioning of fire stations and vehicles in urban municipalities to ensure the entire city is safe in the event of k simultaneous fires.

For the minimum domination problem, a new hybrid approach called IRIS is proposed. IRIS is designed as a general-purpose framework that bridges the gap between exact integer linear programming solvers and heuristic search by iteratively fixing selected variables to reduce the search space. The novelty lies in its flexible subproblem construction mechanism, which can be tailored using various selection strategies. In this study, we implement and evaluate a specific configuration of IRIS that utilizes historical statistical data and a node-coverage-based heuristic to intelligently identify variables for fixing. This targeted approach allows the ILP solver to find high-quality solutions for large-scale instances that are computationally prohibitive for exact methods. Experimental results demonstrate that IRIS achieves competitive performance compared to the best existing methods, establishing it as a valid alternative for solving domination and potentially other NP-hard problems.

Keywords: domination problems on graphs, variable neighborhood search, integer linear programming, hybrid algorithms, large graphs, complex networks

Research area: Computer Science
Research sub-area: Optimization

Садржај

Поглавље 1

Увод

Доминацијски проблеми на графовима су интензивно проучавани, како са теоријске, тако и са практичне стране, пре свега због њихове широке примене у различитим областима. На пример, у бежичним ад-хок и сензорским мрежама, повезани доминирајући скуп чворова може да се користи за формирање виртуелне окоснице мреже (енг. *backbone network*), односно за одређивање чворова који ће бити одговорни за прослеђивање порука у мрежи [115]. У анализи друштвених мрежа, доминирајући скупови могу бити коришћени за поналазак најутицајнијих корисника, односно корисника чије активности допиру до највећег броја других корисника [33]. Доминирајући скупови су коришћени и у анализи управљивости мрежа (енг. *network controllability*) [83], где је показано да се са малим бројем чворова може контролисати цео систем [91], као и у анализи протеинских интеракција, где играју кључну улогу у стабилности мреже и регулаторним функцијама [113]. Поред тога, коришћени су и у различитим аспектима машинског учења, попут кластеровања у графовима [22], као и избору атрибута (енг. *feature selection*) [105]. Доминацијски проблеми налазе примену и у областима као што су телекомуникације [9], распоређивање (енг. *scheduling*) [101], а повезани су и са локацијским проблемима [54].

Поред великог практичног значаја, доминацијски проблеми имају и дубоко развијену теоријску основу. Истраживања доминацијских проблема трају више деценија, а један од централних теоријских циљева било је одређивање горњих и доњих граница доминацијског броја, односно кардиналности оптималног доминирајућег скупа [15, 19, 74, 94]. Границе су испитиване, како за опште графове, тако и за специфичне класе графова, попут бипартитних и хордалних графова, као и усмерених Де Брујинових графова. Поред тога, значајан број радова је посвећен анализи теоријске сложености проблема [28, 34, 81, 51, 41].

Формално, за дати неусмерени граф $G = (V, E)$, доминирајући скуп се дефинише као подскуп чворова S такав да је сваки чвор графа или елемент скупа S или је граном повезан са чвором из S. Циљ је пронаћи такав скуп S минималне кардиналности. С обзиром на то да се доминирајући скупови користе у разним областима, како у научним, тако и у индустријским применама, где свака доноси додатне услове, било је неопходно дефинисати различите варијанте овог проблема. Неке од основних и често разматраних варијанти доминацијског проблема су:

- доминирајући скуп са капацитетима (енг. *Minimum Capacitated Dominating Set*, MCapDS) [92], где сваки чвор има ограничење (капацитет) за број суседа које може доминирати,
- независни доминирајући скуп (енг. *Minimum Independent Dominating Set*, MIDS) [98], где чворови који су у скупу S не смеју да имају суседе који су такође у S,
- тотални доминирајући скуп (енг. *Minimum Total Dominating Set*, MTDS) [62], где сви чворови, укључујући и елементе скупа S, морају да имају бар једног суседа у S,

- повезани доминирајући скуп (енг. *Minimum Connected Dominating Set*, MCDS) [21], где скуп S индукује повезани подграф графа G,
- доминирајући скуп позитивног утицаја (енг. *Minimum Positive Influence Dominating Set*, MPIDS) [20], где бар половина суседа сваког чвора припада скупу S.

Додатно, могу се дефинисати и тежинске варијанте ових проблема, додељивањем тежина чворовима и/или гранама графа. У том случају, циљ је пронаћи доминирајући скуп минималне укупне тежине.

У оквиру ове дисертације биће решавано неколико варијанти доминацијских проблема: проблем минималне тежинске тоталне доминације (енг. *Minimum Weighted Total Dominating Set*, MWTDS), минималне тежинске независне доминације (енг. *Minimum Weighted Independent Dominating Set*, MWIDS), k-јаке римске доминације (енг. *k-strong Roman Dominating Set*, KSRDS), као и основни проблем минималне доминације (енг. *Minimum Dominating Set*, MDS), са посебним фокусом на инстанце великих димензија. Главни циљ рада је развој ефикасних метода за решавање наведених проблема, са посебном пажњом усмереном на велике графове. Дисертација обухвата и егзактне и хеуристичке методе, као и њихове хибридне комбинације, а истраживање је усмерено ка разумевању структуре проблема, побољшању перформанси постојећих алгоритама и примени развијених модела у практичним сценаријима. Конкретни циљеви укључују:

- развој методе променљивих околина за проблем MWTDS и анализа примене у ширењу информација у друштвеним мрежама,
- формулација унапређених ILP модела и похлепног алгоритма за проблем MWIDS,
- развој методе променљивих околина за проблем KSRDS, као и примена на распоређивање ватрогасних станица и камиона у регионима, тако да граф буде безбедан у случају k истовремених пожара,
- развој новог хибридног приступа и примена на проблем MDS на великим графовима.

Сви разматрани проблеми су НП-тешки. Због значаја основне варијанте проблема, у наставку ће бити дат доказ за њу, а за све изведене варијанте се углавном може лако успоставити свођење и показати да су НП-тешке.

1.1 Сложеност решавања проблема минималне доминације

Основни проблем минималне доминације је НП-тежак, што се може показати свођењем проблема покривача грана (енг. *vertex cover*) на њега. Покривач грана је један од 21 проблема за које је Карп првобитно доказао да су НП-комплетни свођењем на проблем задовољивости исказних формула (енг. *satisfiability problem*, SAT) [70]. Овај проблем се дефинише на следећи начин: покривач грана графа G је такав скуп чворова да је свака грана суседна бар једном од чворова из скупа.

Свођење се врши тако што се за сваку грану додаје по један чвор – ако се грана састоји од чворова u и v, додаје се чвор w и гране uw и vw, а оригиналну грану uv задржавамо. Другим речима, уместо сваке гране прави се троугао. Оваква трансформација се може извршити у полиномском времену. Тако добијени граф G' има доминирајући скуп кардиналности k ако и само ако G има покривач грана кардиналности k.

=> Нека је S доминирајући скуп графа G' кардиналности k. S не садржи ниједан од новододатих чворова uv. Ако би садржао такав чвор, он би могао да се замени било чвором u, било чвором v, након чега би скуп S и даље био доминирајући. Како су сви новододати чворови доминирани, важи да скуп S за сваку грану из G садржи бар један њен крај. Стога је S и покривач грана графа G кардиналности k.

<= Нека је сада S покривач грана графа G кардиналности k. По дефиницији покривача грана, свака грана има бар један крај у S. Односно, сваки новододати чвор из G' има суседа из S. Исто важи и за оригиналне чворове, па је S доминирајући скуп графа G' кардиналности k.

Поред тога, одлучива варијанта проблема је НП-комплетна. Она припада класи НП јер је једноставно у полиномском времену верификовати решење, односно проверити да ли је неки подскуп од k чворова доминирајући скуп.

1.2 Методе за решавање оптимизационих НП-тешких проблема

Оптимизациони проблеми се могу у општем облику формулисати као минимизација функције циља уз одређена ограничења. Односно, формално:

$$\begin{array}{ll} \min & f(x) \\ \text{уз ограничења} & g_i(x) \leq b_i, \quad i = 1, 2, \ldots, m \\ & x \in S \end{array}$$

Треба напоменути да је проблем максимизације еквивалентан проблему минимизације, јер се максимизација функције $f(x)$ може извршити минимизацијом функције $-f(x)$. Слично, ограничења у облику $g_i(x) \geq b_i$ се могу изразити као $-g_i(x) \leq -b_i$. Поред тога, једнакосна ограничења се могу представити као пар неједначина, те наведена формулација не губи на општости.

У случају да су функција циља и ограничења линеарни, а скуп $S = \mathbb{R}^n$, добија се проблем линеарног програмирања (енг. *Linear Programming*, LP). Најранији радови на тему линеарне оптимизације датирају још из 19. века када је Фурије објавио методу за решавање система линеарних неједначина [106]. Године 1939. Канторович је формулисао проблеме алокације ресурса као проблеме линеарне оптимизације [66, 35]. Међутим, први практични алгоритам за решавање проблема линеарног програмирања развио је Данциг 1947. године и назвао га је симплекс метода [36]. Овај алгоритам проналази оптимално решење кретањем дуж темена допустивог региона. Како је број темена експоненцијалан у односу на број променљивих, у најгорем случају временска сложеност симплекс методе је експоненцијална [76]. Упркос томе, симплекс метода се у пракси показала веома ефикасном [96]. Први полиномски алгоритам за решавање проблема линеарног програмирања развијен је 1979. године [72]. Значајнија побољшања у пракси донела је метода унутрашње тачке (енг. *interior-point method*) [69], која се заснива на кретању кроз унутрашњост допустивог региона, а такође је полиномске сложености.

Уколико се захтева да неке или све променљиве буду целобројне, добија се проблем целобројног линеарног програмирања (енг. *Integer Linear Programming*, ILP), који је још један од 21 Карпових НП-тешких проблема [70]. Како је овај проблем НП-тежак, сви НП-тешки проблеми могу се свести на њега. Стога је развијено доста напредних техника за његовор решавање, од којих ће неке бити описане у наставку.

Генерално, методе за решавање оптимизационих проблема се могу поделити у две главне групе: егзактне и апроксимативне. Егзактне методе пружају теоријску гаранцију оптималности добијеног решења. Другим речима, уз довољно времена и меморије, проналазе оптимално решење. Са друге стране, апроксимативне методе не гарантују оптималност, али својом ефикасношћу надомешћују овај недостатак. С обзиром на то да су проблеми НП-тешки, егзактне методе често постају неупотребљиве у пракси, поготово на инстанцама великих димензија. Како се у овој дисертацији посебан акценат ставља баш на такве велике инстанце, више пажње ће бити посвећено апроксимативним методама.

1.2.1 Егзактне методе

У класи егзактних метода, између осталих, налазе се динамичко програмирање, гранање са ограничавањем (енг. *branch and bound*, BnB), метода одсецајућих равни, програмирање ограничења, као и A* породица алгоритама [109]. Због њеног значаја у практичним ILP решавачима као што су `Gurobi` и `IBM ILOG CPLEX`, у наставку ће бити описана метода гранања са ограничавањем, као и нека њена унапређења. Ова метода, иницијално предложена 1960. године [77], заснована је на „подели-па-владај“ (енг. *divide-and-conquer*) принципу. Проблем се решава итеративном поделом простора претраге на мање потпросторе (гранање), док се ефикасност постиже одбацивањем (ограничавањем) оних потпростора за које се може доказати да не садрже оптимално решење.

Процес претраге се може представити као дрво претраге, где корен стабла представља иницијални проблем са целим простором допустивих решења. Сваки чвор у стаблу представља један потпроблем дефинисан додавањем нових ограничења на иницијални скуп.

За сваки чвор (потпроблем) решава се његова линеарна релаксација. Линеарна релаксација се добија уклањањем услова целобројности, чиме се омогућава примена ефикасних алгоритама попут симплекс методе. Нека је z_{LP} вредност функције циља релаксираног проблема, а z^* вредност функције циља до тада најбољег пронађеног целобројног решења (енг. *incumbent*). У контексту проблема минимизације (као што је проблем доминације), z_{LP} представља доњу границу (енг. *lower bound*) за тај потпроблем, док z^* представља глобалну горњу границу (енг. *upper bound*).

Потпроблем се одсеца (енг. *pruning*) и даље не разгранава у следећа три случаја:

1. **Недопустивост:** Линеарна релаксација нема решење (простор претраге тог чвора је празан).

2. **Оптималност:** Решење линеарне релаксације је целобројно. Уколико је ово решење боље од тренутног z^*, оно постаје ново z^*.

3. **Ограничавање:** Вредност $z_{LP} \geq z^*$. Како решење релаксације представља доњу границу, односно даје најбољу могућу вредност у том потпростору, јасно је да целобројно решење у тој грани не може бити боље од већ пронађеног.

Уколико ниједан од ових услова није испуњен, врши се **гранање**. Бира се једна променљива x_i која у LP решењу има нецелобројну вредност $\hat{x}_i$, и креирају се два нова потпроблема (чвора деце) додавањем ограничења: $x_i \leq \lfloor \hat{x}_i \rfloor$ и $x_i \geq \lceil \hat{x}_i \rceil$. Избор променљиве за гранање представља један од кључних елемената BnB методе и може значајно утицати на брзину конвергенције алгоритма. Стога су развијене бројне хеуристике за избор променљиве, засноване, између осталог, на децималном делу вредности променљиве у LP решењу, као и историјском утицају променљиве у претходним чворовима [112].

Псеудокод основног алгоритма гранања са ограничавањем приказан је у алгоритму 1.

Алгоритам 1 Основни алгоритам гранања са ограничавањем

Улаз: инстанца ILP проблема P
Излаз: оптимално решење x^*

$\mathcal{L} \leftarrow \{P\}$ ▷ иницијализација листе потпроблема
$z^* \leftarrow \infty$ ▷ глобална горња граница
$x^* \leftarrow$ null
while $\mathcal{L} \neq \emptyset$ **do**
    $P_k \leftarrow$ `selectNode`$(\mathcal{L})$
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \{P_k\}$
    $(x_{LP}, z_{LP}) \leftarrow$ `solveRelaxation`(P_k)
    if `isInfeasible`(x_{LP}) **then**
        continue ▷ одсецање због недопустивости
    end if
    if $z_{LP} \geq z^*$ **then**
        continue ▷ одсецање због границе
    end if
    if `isInteger`(x_{LP}) **then**
        $z^* \leftarrow z_{LP}$
        $x^* \leftarrow x_{LP}$
    else
        $(P_1, P_2) \leftarrow$ `branch`(P_k, x_{LP})
        $\mathcal{L} \leftarrow \mathcal{L} \cup \{P_1, P_2\}$
    end if
end while
return x^*

Савремени решавачи имплементирају читав низ унапређења ове основне методе. Најважнија међу њима је метода **гранања са одсецањем** (енг. *branch and cut*, BnC). Ова метода комбинује BnB са методом равни одсецања (енг. *cutting planes*) [112].

Основна идеја је да се у сваком чвору стабла претраге, након решавања LP релаксације, покуша додати једна или више валидних неједнакости (енг. *valid inequalities*), односно резова (енг. *cuts*). Ове неједнакости „одсецају“ нецелобројно LP решење, а да притом не елиминишу ниједно допустиво целобројно решење.

У основној BnC методи, процес у сваком чвору се одвија на следећи начин:

1. Решава се LP релаксација.

2. Уколико је решење нецелобројно, решава се проблем сепарације како би се пронашле нарушене валидне неједнакости (нпр. Гоморијеви резови, резови покривача, резови клика [112]).

3. Ако су такве неједнакости пронађене, додају се у модел и враћамо се на корак 1 (поновно решавање ојачане релаксације).

4. Уколико резови нису пронађени или је њихов утицај занемарљив, приступа се стандардном гранању као код BnB методе.

Овај приступ омогућава значајно јаче доње границе у чворовима стабла, што драстично смањује број чворова које је потребно експлицитно претражити.

Поред тога, развијене су и друге варијанте и унапређења BnB методе, као што су:

- Метода гранања са оцењивањем (енг. *branch and price*), где се за решавање линеарне релаксације проблема користи метода генерисања колона [10]. Ова метода је посебно корисна код проблема са великим бројем променљивих и структуром погодном за декомпозицију.

- Комбинација позната под називом гранање са одсецањем и оцењивањем (енг. *branch cut and price*) [38], која користи предности оба приступа.

1.2.2 Метахеуристике

Међу апроксимативним алгоритмима, који, као што је већ споменуто, не дају гаранцију оптималности добијеног решења, посебно се издвајају метахеуристике. Код метахеуристичких метода, за разлику од стандардних апроксимативних алгоритама, не постоји ни гаранција максималног одступања од оптималног решења. Упркос томе, ове методе су се показале као веома успешне у пракси, поготово на великим инстанцама НП-тешких проблема. За разлику од хеуристика, изузетно су опште и могу се применити на широк спектар оптимизационих проблема.

У оквиру сваке метахеуристике могу се издвојити два аспекта: експлорација (диверзификација) и експлоатација (интензификација). Експлорација се односи на истраживање нових региона простора претраге, док се експлоатација односи на фокусирање на регионе који су већ истражени и у којима се налази боље решење. Како би се показало да су обе ствари неопходне, могу се размотрити две крајности:

- Само експлорација – ако би били истраживани само нови региони простора претраге, практично би била извршена случајна претрага.

- Само експлоатација – уколико би се увек ишло ка најбољем решењу у региону из ког је претрага започета, дошло би до заглављивања у локалном оптимуму.

Дакле, потребно је успоставити баланс између ова два супротстављена аспекта.

На основу броја решења које користе у току претраге, метахеуристике се могу поделити у две главне групе: популационе метахеуристике и метахеуристике вођене једним решењем. Популационе метахеуристике користе скуп решења која утичу једна на друге на неки начин, док метахеуристике вођене једним решењем користе само једно решење током читаве претраге. У наставку ће укратко бити описане неке од чешће коришћених, а за детаљнији приказ видети [109, 49].

Метахеуристике вођене једним решењем

Алгоритам који у сваком кораку претраге иде ка најбољем решењу у околини тренутног решења назива се локална претрага. У претходној секцији је већ показано да нема смисла самостално користити овакав приступ (само интензификација без диверзификације) за решавање НП-тешких оптимизационих проблема. Међутим, локална претрага може да буде изузетно корисна као градивни елемент разних метахеуристика, обично оних вођених једним решењем, али може се користити и унутар популационих. Постоје две основне врсте локалне претраге:

- Локална претрага са првим побољшањем (енг. *first improvement*) – пролази кроз решења у околини тренутног решења у случајном редоследу и бира прво које је боље од тренутног.

- Локална претрага са најбољим побољшањем (енг. *best improvement*) – у сваком кораку претраге се бира најбоље од свих решења у околини тренутног решења.

Симулирано каљење (енг. *Simulated Annealing*) [75] је метахеуристика која се заснива на аналогији са процесом каљења метала, где се метал загрева на одређену температуру и онда постепено хлади са циљем побољшања његових физичких карактеристика. У контексту оптимизације, као и код локалне претраге, када се у околини тренутног решења пронађе боље, прелази се на њега. Са друге стране, како бисмо избегли заглављивање у локалном минимуму, примењује се диверзификација – са одређеном вероватноћом прихвата се и лошије решење. Та вероватноћа опада током итерација, слично као температура у поступку стварног каљења. Основна идеја је да на почетку рада алгоритма, када још увек није пронађено висококвалитетно решење, буде омогућена већа диверзификација како би се истражио што већи део простора претраге. Касније, у току рада алгоритма, када је вероватно већ достигнуто неко квалитетно решење, фокус се преусмерава на интензивније истраживање његове околине како би се додатно унапредио, и избегло удаљавање од већ пронађених добрих решења.

Критеријум заустављања алгоритма, а и свих других метахеуристика, може бити неки од следећих услова: укупан број итерација, протекло време (у секундама), проналазак довољно доброг решења, број узастопних итерација без побољшања. Наравно, може се користити и адекватна комбинација више услова.

Итерирана локална претрага (енг. *Iterated Local Search*, ILS) [86] се заснива на једноставној идеји итеративног покретања локалне претраге на промењеном тренутном решењу. Промена која се уводи назива се и пертурбација, и углавном се реализује случајним одабиром суседа који је значајно даље од тренутног решења у односу на суседе који се разматрају у току локалне претраге. Након локалне претраге на промењеном решењу, бира се да ли ће се оно прихватити као ново тренутно решење. Ова одлука може бити једноставна – ново решење се прихвата ако је боље од тренутног, а може бити и сложенија – нпр. прихватање новог решења зависи од историјата претраге.

Табу претрага (енг. *Tabu Search*, TS) [50, 48] је метахеуристика која користи меморију како би избегла заглављивање у локалном оптимуму. Наиме, у току рада алгоритма одржава се табу листа – скуп претходних решења (или делова решења) која су забрањена у наредним корацима претраге, са циљем избегавања цикличног понашања. Дужина табу листе представља важан параметар алгоритма – уколико је превелика, може доћи до претеране диверзификације и изостајања интензификације, док ако је премала, може доћи до преране конвергенције ка субоптималном решењу. Стога се овај параметар може и динамички мењати током рада алгоритма. Поред тога, табу претрага користи и концепт аспирације, који омогућава да се табу ограничења превазиђу уколико ново решење испуњава неки додатни критеријум, попут надмашивања тренутно најбољег решења.

Похлепна стохастичко-адаптивна претрага (енг. *Greedy Randomized Adaptive Search Procedure*, GRASP) [43] је метахеуристика која се састоји из два главна дела: конструкције решења и побољшања решења. Конструкција решења је заснована на похлепном приступу, али са додатком стохастичког елемента. У сваком кораку конструкције решења, на насумичан начин се бира елемент из скупа најбољих кандидата, који се затим додаје у тренутно решење. У случају да је број кандидата једнак један, овај приступ се своди на класичну похлепну хеуристику. Након тога, на добијено решење се примењује локална претрага са циљем његовог унапређења. Читав алгоритам се састоји од итеративног понављања ове две фазе до испуњења критеријума заустављања, а крајње решење је најбоље пронађено током свих итерација.

Метода променљивих околина (енг. *Variable Neighborhood Search*, VNS) [88] се састоји од два главна дела: размрдавања (енг. *shaking*) (задужено за диверзификацију) и локалне претраге (као и обично, задужена за интензификацију). Размрдавањем се на насумичан начин добија ново решење које се налази у k-тој околини тренутног решења. Потом се на новодобијено решење примењује локална претрага како бисмо га унапредили.

Овај процес се понавља док се не испуни критеријум заустављања. Пре свега, потребно је дефинисати више околина, од којих је свака следећа шира од претходне. Алгоритам прво покушава да нађе боље решење од тренутног у првој околини, па у случају да не успе, прелази на следећу. Ако је боље решење пронађено, оно се прихвата и поступак се понавља почевши од прве околине. У случају да је нађено решење истог квалитета као и тренутно, оно се прихвата са вероватноћом одређеном параметром $moveProb$. Како је метода променљивих околина коришћена за решавање више доминацијских проблема у оквиру ове дисертације, дат је и псеудокод алгоритма 2.

Алгоритам 2 Основна метода променљивих околина

Улаз: инстанца проблема I, $k_{\min}, k_{\max} > 0$, скуп околина $\mathcal{N}_{k_{\min}}, \ldots, \mathcal{N}_{k_{\max}}$, функција f, $moveProb$
Излаз: решење s

```
1: s ← initialize(I)
2: while ! terminationCriteriaSatisfied() do
3:     k ← k_min
4:     while k ≤ k_max do
5:         s' ← shake(s, N_k)
6:         s'' ← localSearch(s')
7:         if f(s'') < f(s) or (f(s'') = f(s) and r ~ U(0,1) < moveProb) then
8:             s ← s''
9:             k ← k_min
10:        end if
11:        k ← k + 1
12:    end while
13: end while
14: return s_best
```

Постоје различите варијације овог алгоритма, међу којима су и:

- редукована метода променљивих околина (енг. *Reduced VNS*, RVNS) [58], где се користи само размрдавање, без локалне претраге,
- општа метода променљивих околина (енг. *General VNS*, GVNS) [58], у којој се уместо локалне претраге користи друга метода променљивих околина,
- накошена метода променљивих околина (енг. *Skewed VNS*, SVNS) [58], која у размрдавању узима у обзир и удаљеност новог решења од тренутног решења,
- метода променљивих околина са декомпозицијом (енг. *Decomposition VNS*, DVNS) [58], која декомпонује решење пре примене локалне претраге.

Популационе метахеуристике

Популационе метахеуристике су углавном инспирисане понашањем животиња или процесима у природи. Основна идеја је да се у сваком кораку претраге користи скуп решења, која међусобно комуницирају на неки начин и тако обогаћују претрагу.

Генетски алгоритам (енг. *Genetic Algorithm*, GA) [61] спада у већу групу еволутивних алгоритама који су осмишљени по угледу на биолошку еволуцију. Користи популацију јединки, од којих свака представља једно решење проблема и има свој код и степен прилагођености (енг. *fitness*). Идеја је да, као и код природне селекције, најприлагођеније

јединке опстају. Еволуција популације се реализује путем оператора селекције, укрштања и мутације.

Задатак селекције је одабир јединки које ће учествовати у укрштању. Постоје различите методе селекције, од којих су неке од најчешће коришћених: рулетска и турнирска селекција. Код рулетске селекције, свака јединка добија вероватноћу избора пропорционалну свом степену прилагођености. С обзиром на то да је могућ велики дисбаланс у степенима прилагођености, што може довести до преране конвергенције, често се користи нормализација или скалирање вредности прилагођености пре примене ове методе, нпр. коришћењем рангова уместо апсолутних вредности. Турнирска селекција се заснива на насумичном одабиру неколико јединки из популације, након чега се као победник бира она јединка која има највећи степен прилагођености.

Укрштање представља процес формирања нових јединки комбиновањем генетског материјала родитеља. Постоје различити начини укрштања, у зависности од начина на који је решење кодирано. Неки од често коришћених су n-позиционо укрштање и униформно укрштање. Код n-позиционог укрштања, насумично се бира n позиција у генетском коду родитеља, након чега се генетски материјал укршта на тим позицијама. Са друге стране, код униформног укрштања, за сваку позицију у генетском коду се насумично бира да ли ће тај део генетског материјала бити преузет од првог или другог родитеља.

Мутација представља случајну промену генетског материјала јединке и уобичајено се примењује са малом вероватноћом након укрштања. Ова операција је задужена за увођење диверзификације у популацију.

Након примене ових оператора, потребно је формирати нову популацију. То се углавном реализује потпуном или делимичном заменом старе популације.

Еволутивне стратегије (енг. *Evolution Strategies*, ES) [102, 16] су још један пример еволутивних алгоритама. Углавном користе за оптимизацију непрекидних функција и разликују се од генетских алгоритама у неколико аспеката. Код еволутивних стратегија се не користи укрштање у класичном смислу, већ се нове јединке формирају применом мутације на родитеље. Мутација се уобичајено изводи додавањем нормално расподељеног шума на вредности променљивих. Битна разлика у односу на генетске алгоритме је у томе што еволутивне стратегије користе адаптивне параметре мутације, који се такође еволуирају током рада алгоритма.

Диференцијална еволуција (енг. *Differential Evolution*, DE) [108] је још један популациони еволутивни алгоритам, који се примарно користи за оптимизацију непрекидних функција. Разлика у односу на генетске алгоритме је у редоследу примене оператора. Наиме, код диференцијалне еволуције се прво врши мутација, па онда укрштање. Мутација се састоји од генерисања нових јединки на основу разлике између насумичних јединки из тренутне популације.

Оптимизацијаројем честица (енг. *Particle Swarm Optimization*, PSO) [71] је пример алгоритма из области интелигенција група (енг. *Swarm Intelligence*)[1]. Код алгоритама из ове групе, свака појединачна јединка је једноставна, али читава група колективно се понаша на неки начин интелигентно. У случају оптимизацијеројем честица, свака честица представља једно решење проблема и креће се кроз простор претраге. Притом, свака честица зна најбоље решење до кога је стигла, као и најбоље решење које је пронашао читав рој. Може се прилагодити и за дискретне проблеме, али како се превасходно користи за непрекидну оптимизацију, биће дат кратак опис овог случаја. Брзина, тј. вектор помераја честице се у свакој итерацији рачуна сабирањем три компоненте:

- инерција – претходна брзина,

[1]Буквалан, и често коришћен, превод би био интелигенцијаројева. Међутим, са развојем ове области и порастом броја различитих алгоритама који нису засновани наројевима инсеката, уведен је општији термин интелигенција група.

- когнитивна брзина – вектор од тренутне позиције честице до позиције најбољег решења до ког је она стигла,
- социјална брзина – вектор од тренутне позиције честице до позиције најбољег решења до ког је читав рој стигао.

Оптимизација колонијом мрава (енг. *Ant Colony Optimization*, ACO) [40] је метахеуристика инспирисана понашањем мрава у природи. Мрави при кретању остављају феромоне на тлу, а остали мрави бирају куда ће ићи на основу количине феромона која се ту налази. Што више мрава иде истим путем, биће више феромона. Са друге стране, феромони током времена испаравају, па што је пут дужи, биће више времена да феромони испаре. На овај начин мрави проналазе најкраћи пут до хране. Како је у случају оптимизације циљ мрава проналазак оптимума, количина феромона се одређује на основу квалитета пронађеног решења. Природна примена овог алгоритма је у проблемима проналаска најкраћег пута, као што је проблем трговачког путника (енг. *Travelling Salesman Problem*, TSP).

1.3 Нотација

Сви решавани проблеми су дефинисани на простим неусмереним графовима, те ће у наставку укратко бити дефинисани основни појмови из теорије графова који ће бити коришћени у наредним поглављима. Неусмерени граф се означава са $G = (V, E)$, где је V скуп чворова, а E скуп грана. Свака грана $e \in E$ одговара пару чворова $e = \{u, v\}$. За два чвора u и v се каже да су суседи уколико постоји грана $e = \{u, v\}$ која их повезује. За сваки чвор $v \in V$, са $N(v)$ се означава (отворена) околина чвора v, односно скуп чворова који су му суседни, тј. формално $N(v) = \{u \in V \mid \{u, v\} \in E\}$. Кардиналност овог скупа се назива степен чвора $d(v) = |N(v)|$. Са $N[v]$ се означава затворена околина чвора v, односно скуп $N(v) \cup \{v\}$. Ако је S подскуп чворова графа, онда се са $G[S] = (S, E[S])$ означава индуковани подграф графа G на чворовима из S, где је $E[S] = \{\{u, v\} \in E \mid u, v \in S\}$.

Поглавље 2

Проблем минималне тежинске тоталне доминације

Тотална доминација, као што је поменуто у уводном поглављу, дефинише се додатним ограничењем у односу на класичну доминацију – поред чворова ван решења, и чворови из решења морају имати бар једног суседа у решењу. Ово је добро изучен проблем, како са практичне, тако и са теоријске стране, а првобитно је уведен у раду [30]. Проблем је НП-тежак, а доказ се може пронаћи у књизи [59]. Овај проблем се природно проширује додељивањем тежина чворовима и гранама ради бољег моделовања реалних проблема. На тај начин се долази до проблема минималне тежинске тоталне доминације (MWTDS), који је предмет овог поглавља, а уведен је у раду [87]. Резултати представљени у овом поглављу објављени су у раду [68].

2.1 Дефиниција проблема

Овај проблем се дефинише на неусмереним графовима $G = (V, E, f_1, f_2)$, где су, као и обично, V и E скупови чворова и грана графа, а $f_1 : V \to \mathbb{R}^+$ и $f_2 : E \to \mathbb{R}^+$ функције које додељују тежине чворовима и гранама, респективно. Ради једноставније нотације, у наставку ће тежина чвора v бити означавана са w_v, а тежина гране $e = \{u, v\}$ са w_e или w_{uv}. Циљ проблема MWTDS је проналазак скупа $S \subseteq V$ који испуњава следећа два услова:

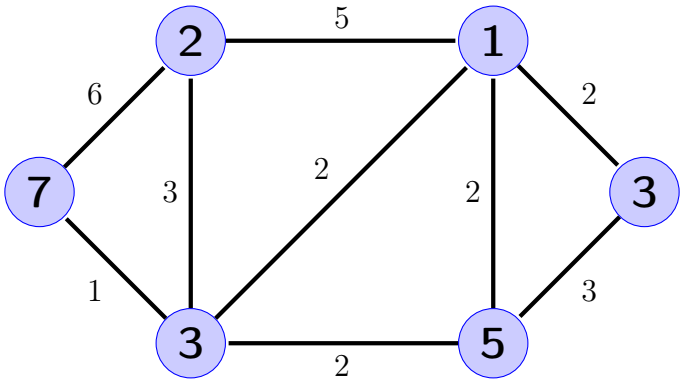
1. S је тотални доминирајући скуп графа G, односно $\forall v \in V, |N(v) \cap S| \geq 1$;
2. S има најмању вредност функције циља која се дефинише као:

$$f(S) = \sum_{v \in S} w_v + \sum_{e \in E[S]} w_e + \sum_{v \in V \setminus S} min\{w_{vu} \mid u \in S\} \tag{2.1}$$

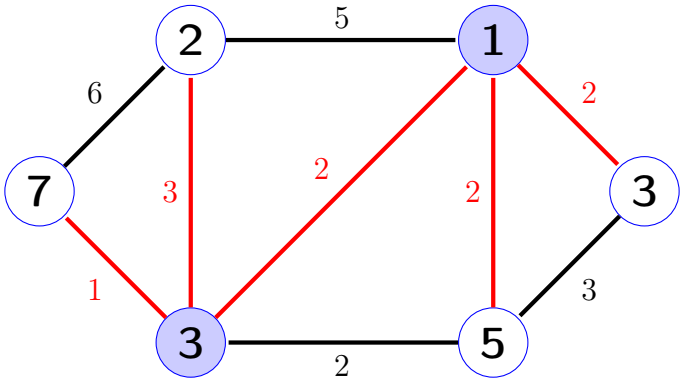
где је $E[S]$ скуп грана подграфа индукованог скупом S. Пратећи номенклатуру из рада [6], прва сума ће бити означавана као цена чворова, друга сума као цена унутрашњих грана, а трећа сума као цена спољашњих грана. Дакле, унутрашње гране су оне које повезују два чвора која су оба у решењу, док су спољашње гране оне које повезују чвор ван решења са чвором из решења. Притом, ако чвор ван решења има више суседа у решењу, за спољашњу грану се бира она која има најмању тежину. Другим речима, спољашња грана је она којом је најјефтиније покрити чвор ван решења.

2.1.1 Пример инстанце

Пример инстанце проблема MWTDS приказан је на слици 2.1. На овој слици, тежине чворова су означене бројевима унутар кругова, док су тежине грана означене на линијама које представљају гране. Оптимално решење проблема MWTDS за ову инстанцу је скуп



Слика 2.1: Пример инстанце проблема MWTDS



Слика 2.2: Оптимално решење проблема MWTDS

чворова обојених у плаво на слици 2.2, чија је вредност функције циља 14. До ове вредности функције циља се долази на следећи начин:

- цена чворова је 4, јер су у решењу чворови са тежинама 1 и 3;
- цена унутрашњих грана је 2, постоји само једна унутрашња грана са тежином 2;
- цена спољашњих грана је 8, јер гране најмање тежине које повезују чворове ван, са онима у решењу, имају тежине: 1, 3, 2 и 2.

2.2 Преглед литературе

Постоји неколико ILP модела који су предложени за проблем MWTDS. У раду [87] уведено је три ILP модела - један основни и два унапређена, заснована на елиминацији променљивих и додавању нових ограничења. Два побољшана модела су представљена у раду [6], где је предложен и приступ заснован на методи гранања са одсецањем који користи валидне неједнакости (уз помоћ Бендерове декомпозиције), као и почетне и прималне хеуристике. Поред тога, у истом раду предложене су и две метахеуристике - похлепна стохастичко-адаптивна метода претраге (енг. *Greedy Randomized Adaptive Search Procedure*, GRASP), самостално, и у комбинацији са генетским алгоритмом.

Перформансе свих ових алгоритама су тестиране на мањим инстанцама проблема (до 125 чворова), на којима се егзактне методе показују добро и успевају да пронађу оптимална решења на већини инстанци. Дакле, потребно је увести нове веће инстанце да би се боље проценио квалитет постојећих алгоритама. Такође, постоји и недостатак хеуристичких приступа који се генерално боље понашају на већим инстанцама.

2.3 Метода променљивих околина за проблем MWTDS

Метода променљивих околина, укратко описана у уводном поглављу, показала је добар учинак на разним НП-тешким оптимизационим проблемима [55]. Псеудокод VNS-а за проблем MWTDS дат је у алгоритму 3. У наставку овог одељка биће описани детаљи дизајна и имплементације предложеног алгоритма.

Алгоритам 3 VNS за проблем MWTDS

Улаз: $G = (V, E, f_1, f_2)$: инстанца проблема; $\kappa_{\min}$: минимална величина околине; $\kappa_{\max}$: максимална величина околине; it_{max}: максималан број итерација; t_{max}: максимално време извршавања у секундама.
$S \leftarrow$ `random`()
$it \leftarrow 0$
$\kappa \leftarrow \kappa_{min}$
while $it < it_{max} \wedge t_{elapsed} < t_{max}$ **do**
    $S_{new} \leftarrow$ `shaking`(S, κ, G)
    `localSearch1`(S_{new}, G)
    if S not improved recently $\wedge$ `localSearch2` not called recently **then**
        `localSearch2`(S_{new}, G)
    end if
    $RV \leftarrow \mathcal{U}(0, 1)$
    if $fit(S_{new}, G) < fit(S, G) \ \vee (fit(S_{new}, G) = fit(S, G) \ \wedge RV < 0.5)$ **then**
        $S \leftarrow S_{new}$
        $\kappa \leftarrow \kappa_{min}$
    else
        $\kappa \leftarrow \kappa + 1$
        if $\kappa \geq \kappa_{max}$ **then**
            $\kappa \leftarrow \kappa_{min}$
        end if
    end if
    $it \leftarrow it + 1$
end while
Излаз: S

2.3.1 Репрезентација решења и иницијализација

Решење је представљено природно, као скуп чворова. Треба напоменути да остали аспекти функције циља, унутрашње и спољашње гране, нису променљиве одлучивања, већ се израчунавају на основу овог решења.

Иницијално решење се конструише на насумичан начин. Сваки чвор има вероватноћу 0.2 да буде укључен у решење, где је ова вредност одређена емпиријски.

2.3.2 Функција прилагођености

Функција прилагођености је дефинисана на следећи начин:

$$fit(S) = viol(S) + \frac{obj(S)}{W_{tot} + 1} \tag{2.2}$$

где је $viol(S)$ број чворова који нарушавају услов тоталне доминације, односно број чворова без иједног суседа у S, $obj(S)$ вредност функције циља из једначине 2.1, а W_{tot} збир тежина свих чворова и грана у графу. Овако дефинисан W_{tot} представља тривијалну горњу границу за квалитет решења проблема. Стога се вредност $\frac{obj(S)}{W_{tot}+1}$ увек налази у интервалу $(0, 1)$, па је $viol(S)$ важнији део функције прилагођености.

Код свих допустивих решења вредност $viol(S)$ је 0, па је вредност функције прилагођености увек мања од 1. Са друге стране, за сва недопустива решења вредност $viol(S)$ је већа од 0, па је вредност функције прилагођености увек већа од 1. Дакле, свако допустиво решење има строго мању вредност функције прилагођености од сваког недопустивог решења. Због овог корисног својства овако дефинисане функције прилагођености, омогућен је суптилан пролазак и кроз допустиве, и кроз недопустиве регионе простора претраге.

2.3.3 Размрдавање

Размрдавање, задужено за диверзификацију у алгоритму, у околини κ реализује се избацивањем κ насумичних чворова из решења. Будући да, углавном, поготово у каснијим фазама рада алгоритма, неће постојати чворови у решењу који су „бескорисни“, тј. не покривају ниједан чвор, решење S_{new} добијено на овај начин ће у већини случајева имати лошију вредност функције прилагођености од S, односно размрдавањем се може од претходно допустивог добити недопустиво решење. С обзиром на то, потребно је након размрдавања применити локалну претрагу да би се добило допустиво, и, потенцијално, боље решење.

2.3.4 Локална претрага

Интензификација је реализована комбинацијом две локалне претраге са првим побољшањем – бржа `localSearch1` (LS1) са временском сложеношћу $O(|V|)$ по итерацији, и спорија, али моћнија `localSearch2` (LS2) са временском сложеношћу $O(|V|^2)$ по итерацији. Детаљи имплементације су приказани у псеудокодовима 4 и 5.

LS1 користи оператор 1-инверзије до првог побољшања, односно покушава се избацивање било ког чвора из решења или додавање чвора ван решења. Као и обично код првог побољшања, постоји пристрасност у редоследу прихватања бољих решења. Како би се оваква пристрасност избегла, на почетку сваке итерације се одређује насумична пермутација чворова. Потом се итерира кроз тако одређен редослед чворова и за сваки чвор се рачуна функција прилагођености решења у ком је тај чвор избачен (ако је претходно био део решења) или додат (уколико је до сада био ван решења). Кад год дође до побољшања, односно смањења вредности функције прилагођености, промена се прихвата и излази се из унутрашње петље, а у супротном се прелази на следећи чвор.

С обзиром на то да је промена у виду додавања или избацивања једног чвора локална по својој природи, нема потребе за рачунањем вредности функције прилагођености од нуле након сваке промене. Уместо тога, вредност функције прилагођености се рачуна на основу претходне вредности и промене која је настала. Ово се постиже инкременталним (парцијалним) рачунањем вредности функције прилагођености, што је детаљно описано у одељку 2.3.5.

LS2 користи стратегију 1-замене до првог побољшања. Систематично се пролази кроз све парове чворова од којих је један у решењу, а један ван, и проверава се ефекат њихове замене, односно избацивања једног и убацивања другог чвора. Поред тога, остављена је могућност и за само додавање, то јест само избацивање неког чвора. Показало се да је у неким случајевима оваква промена ефективнија од замене, иако је замена углавном

моћнија (мање локална по природи). Као и код LS1, користи се инкрементално рачунање вредности функције прилагођености како би се убрзало извршавање. Међутим, то није довољно да надомести већу временску сложеност овог алгоритма, па се LS2 не примењује увек, већ само повремено. Конкретно, користи се само ако су наредна два услова испуњена:

- решење се није побољшало у последњих 100 VNS итерација;
- LS2 није позвана у последњих 10 VNS итерација.

Алгоритам 4 Функција localSearch1

```
1: Улаз: S: решење; G = (V, E, f1, f2): инстанца проблема.
2: improved ← true
3: currFit ← fit(S, G)
4: while improved do
5:     improved ← false
6:     for v ∈ randomOrder(V) do
7:         if v ∈ S then
8:             newFit ← recalcNodeRemoved(S, v, G)
9:             if newFit < currFit then
10:                S ← S \ {v}
11:                currFit ← newFit
12:                improved ← true
13:                break
14:            end if
15:        else                                          ▷ v ∈ V \ S
16:            newFit ← recalcNodeAdded(S, v, G)
17:            if newFit < currFit then
18:                S ← S ∪ {v}
19:                currFit ← newFit
20:                improved ← true
21:                break
22:            end if
23:        end if
24:    end for
25: end while
26: Излаз: S
```

2.3.5 Инкрементално рачунање вредности функције прилагођености

Као што је већ поменуто, промена која настаје додавањем једног чвора v у решење је локалне природе. Оваква операција утиче само на сам чвор v и његове суседе. Вредност функције прилагођености се може ажурирати на следећи начин:

1. Тежина чвора v се додаје на вредност функције прилагођености, јер је он додат у решење.
2. Ако v има суседа у решењу, тежина његове екстерне гране се одузима.
3. Тежине интерних грана чвора v се додају.

Алгоритам 5 Функција localSearch2

```
1: Улаз: S: решење; G = (V, E, f1, f2): инстанца проблема.
2: improved ← true
3: currFit ← fit(S, G)
4: while improved do
5:     improved ← false
6:     for v_out ∈ randomOrder(V \ S) do
7:         newFit ← recalcNodeAdded(S, v_out, G)
8:         if newFit < currFit then          ▷ додавање без избацивања већ побољшава
9:             S ← S ∪ {v_out}
10:            currFit ← newFit
11:            improved ← true
12:            break
13:        end if
14:    end for
15:    if improved then
16:        continue
17:    end if
18:    for v_in ∈ randomOrder(S) do
19:        newFit ← recalcNodeRemoved(S, v_in, G)
20:        if newFit < currFit then          ▷ избацивање без додавања већ побољшава
21:            S ← S \ {v_in}
22:            currFit ← newFit
23:            improved ← true
24:            break
25:        end if
26:        S ← S \ {v_in}                                          ▷ избацивање
27:        for v_out ∈ randomOrder(V \ S) do          ▷ провера ефекта додавања
28:            newFit ← recalcNodeAdded(S, v_out, G)
29:            if newFit < currFit then   ▷ избацивање и додавање (замена) побољшава
30:                S ← S ∪ {v_out}
31:                currFit ← newFit
32:                improved ← true
33:                break
34:            end if
35:        end for
36:        if improved then
37:            break
38:        else
39:            S ← S ∪ {v_in}                  ▷ враћање претходно избаченог v_in
40:        end if
41:    end for
42: end while
43: Излаз: S
```

4. За сваког суседа u чвора v који нема суседа у решењу, број чворова који нарушавају услов тоталне доминације се смањује за један (зато што чвор u сада има суседа у решењу). Додатно, ако u није у решењу, додаје се тежина његове екстерне гране (ако u јесте у решењу, ова грана је у ствари интерна и њена тежина је већ додата у кораку

3).

5. За сваког суседа чвора v ажурира се његова тежина екстерне гране – проверава се да ли је чвор v постао његов најближи сусед у решењу.

Потпуно аналогно резоновање важи и за случај када се чвор избацује из решења. Псеудокодови ове две функције су дати у додатку А.

Да би ова процедура могла да се извршава ефикасно, потребно је чувати додатне информације о стању решења у адекватној структури података. Конкретно, за сваки чвор у графу се чува скуп грана које га повезују са суседима који су у решењу. Током читавог рада алгоритма, ови скупови се одржавају уређени по тежини грана. На овај начин је омогућен $O(1)$ приступ екстерног грани било ког чвора.

2.4 Експериментални резултати

У овом одељку су представљни емпиријски докази квалитета предложеног алгоритма. Упоређени су сви релевантни приступи за решавање проблема MWTDS који су доступни у литератури на свим постојећим инстанцама проблема, а уведене су и неке нове. Разматране су следеће четири методе:

1. F2 ILP модел који су предложили Алварез-Миранда и Синл [6], овде означен као Ilp. Треба напоменути да је у истом раду предложен и F1 ILP модел, али је показано да је F2 модел бољи на њиховом скупу инстанци (његовим решавањем је више инстанци решено до оптималности). Поред тога, исти аутори су комбиновали F1 и F2 моделе са радним оквиром заснованим на гранању са одсецањем, добивши тако методе под називом $F1^+$ и $F2^+$. Ове унапређене верзије ($F1^+$ и $F2^+$) остварили су боље резултате од основних ILP модела (F1 и F2) на малим и средње великим инстанцама. Међутим, због недостатка информација није их било могуће поново имплементирати, нити су њихове извршне верзије биле доступне. Како би поређење било што поштеније, у поређење су укључени најбољи резултати свих ових метода, када год је то могуће. Свакако метода гранања са одсецањем припада класи егзактних метода, што је чини тешко применљивом на великим инстанцама које су уведене у овом раду. Стога верујемо да се перформансе F2 модела не би много разликовале од оригиналног $F2^+$ модела на великим инстанцама.

2. Похлепна стохастичко-адаптивна метода претраге (енг. *Greedy Randomized Adaptive Search Procedure*, GRASP) која је предложена у раду [6], означена као Grasp.

3. Хибрид Grasp-а и генетског алгоритма, означен као Grasp+Ga, такође предложен у раду [6].

4. Предложени VNS алгоритам описан у одељку 2.3, означен као Vns.

Сви алгоритми су имплементирани у програмском језику Python и извршавани помоћу PyPy интерпретера, који представља бржу алтернативу стандардном Python интерпретеру CPython. Експерименти су извођени на рачунару са Intel Core i9-9900KF процесором са 3.6GHz, уз ограничење меморије на 6GB по покретању, под оперативним системом Microsoft Windows 10 Pro. Како ни изворни код, ни извршне верзије алгоритама из литературе нису биле доступне, Grasp и Grasp+Ga су пажљиво поново имплементирани. ILP модели су решавани коришћењем `CPLEX` 20.1 решавача. Максимално дозвољено време за решавање сваке инстанце је било постављено на 30 минута (1800 секунди), осим за инстанце друштвених мрежа из одељка 2.5 где је коришћено дуже временско ограничење.

Ради фер поређења и конзистентности са методологијом успостављеном у релевантној литератури [6, 87], сви алгоритми извршавани су тачно једном по инстанци.

2.4.1 Инстанце проблема

Коришћена су два скупа случајно генерисаних инстанци из литературе. Први скуп означен као `MA` [87] садржи 45 инстанци величине између 20 и 100 чворова. Други скуп означен као `AMS` [6] састоји се од 125 инстанци од којих највеће имају 125 чворова. Генерисане су коришћењем Ердош-Ренји модела [42] $G(n, p)$, где генерисани граф има n чворова и вероватноћу p за постојање гране између свака два чвора. Да би се испитао утицај различитих тежина чворова и грана на рад алгоритама, тежине чворова, односно грана, додељиване су на случајан начин из униформне расподеле $\mathcal{U}\{a, b\}$, односно $\mathcal{U}\{c, d\}$, где су $(a, b, c, d) \in \{(1, 50, 1, 10), (1, 25, 1, 25), (1, 10, 1, 50)\}$. На овај начин постоји три различите групе инстанци са различитим односом тежина чворова и грана. У првој групи су тежине чворова веће од тежина грана, а самим тим су и битније за квалитет решења. Друга група се састоји од инстанци код којих су тежине чворова и грана у истом распону, па су једнако битне. Коначно, трећа група садржи инстанце код којих су тежине грана веће од тежина чворова, па су гране битније за квалитет решења.

Како је у овом раду главни фокус на поређењу хеуристичких метода, поред ова два скупа инстанци мале и средње величине, претходно описаном методологијом генерисан је и нови скуп великих инстанци означен са `NEW`. Број чворова у овом скупу је из скупа $\{250, 500, 1000\}$, док је вероватноћа постојања гране између два чвора $p \in \{0.2, 0.5, 0.8\}$. Пошто је генерисано по 5 графова за сваку од 27 различитих конфигурација $\{(1, 50, 1, 10), (1, 25, 1, 25), (1, 10, 1, 50)\} \times \{250, 500, 1000\} \times \{0.2, 0.5, 0.8\}$, укупан број инстнаци у скупу `NEW` износи 135.

Јавни репозиторијум са (ре)имплементацијом сва четири метода која су поређена, свим инстанцама проблема, сировим резултатима извршавања, као и додатним материјалима доступан је на адреси `https://github.com/StefanKapunac/wtdp_public`.

2.4.2 Параметри алгоритма

Спроведено је више прелиминарних експеримената како би се утврдиле адекватне вредности контролних параметара Vns алгоритма. Као резултат, следеће вредности су одабране као најбоље:

- $\kappa_{\min} = 1$;
- $\kappa_{\max} = \min\{20, \frac{|V|}{5}\}$;
- $it_{\max} = 3900$.

Параметар $it_{\max}$ постављен је на 3900 ради фер поређења са најбољом хеуристичком методом из литературе, што је Grasp+Ga алгоритам. Почетна популација од 100 јединки генерисана је употребом Grasp алгоритма. Након тога, у свакој од 20 итерација генетског алгоритма, сви парови јединки из тренутне популације су коришћени за генерисање нове популације. Како је величина популације била 20, укупан број могућих парова јединки износио је $\frac{20 \cdot 19}{2} = 190$. Стога је укупан број итерација алгоритма Vns једнак укупном броју корака алгоритма Grasp+Ga и износи $100 + 20 \cdot 190 = 3900$.

Избор $\kappa_{\min} = 1$ је разуман узевши у обзир величину разматраних инстанци и задато временско ограничење. Веће вредности $\kappa_{\min}$ доводе до јачих промена у фази размрдавања, што може проузроковати нестабилност и спорију конвергенцију. Са друге стране, веће

вредности $\kappa_{\min}$ могу бити корисне у раду са већим графовима, као што је случај са инстанцама друштвених мрежа анализираним у одељку 2.5.

За подешавање параметра $\kappa_{\max}$ коришћен је следећи скуп од дванаест конфигурација: $\{\min\{x,y\}|(x,y) \in \{10,20,40\} \times \{logn, \sqrt{n}, \frac{n}{5}, \frac{n}{2}\}\}$. Процес подешавања извршен је на подскупу инстанци, тако што је из сваке групе инстанци одабран по један представник (од укупно пет). Будући да је укупан број инстанци једнак 315, за подешавање овог параметра коришћен је подскуп од $\frac{315}{5} = 63$ инстанце. Најбољи резултати, у погледу просечног квалитета решења, на овом подскупу инстанци добијени су за конфигурацију $x = 20$ и $y = \frac{n}{5}$, те су ове вредности коришћене у коначној евалуацији алгоритма VNS.

2.4.3 Опис резултата

Сумарни резултати су приказани у табелама 2.1 и 2.2. Табела 2.1 приказује резултате метода ILP и VNS на инстанцама из скупа `MA`. Прва колона представља групу инстанци; на пример, `MA`-20 означава групу инстанци са $|V| = 20$ из референтног скупа `MA`. Друга колона садржи број инстанци у датој групи, док трећа колона ($\overline{best}$) приказује просечан квалитет најбољег решења, узимајући у обзир сва доступна решења из литературе, укључујући и резултате предложеног VNS алгоритма.

Следећа два блока приказују резултате метода ILP и VNS. Први блок се састоји од две колоне: колона $\overline{obj}$ представља просечан квалитет решења за све инстанце у групи, док колона $\#opt$ приказује број инстанци које су оптимално решене. Други блок, који се односи на VNS, садржи четири колоне: колона $\bar{t}$ означава просечно време извршавања у секундама (усредњено по свим инстанцама у групи), $\overline{obj}$ даје просечан квалитет решења, $\overline{pg\%}$ представља просечну релативну разлику решења у односу на најбоље решење (у процентима), тј. $(\frac{obj-best}{best}) \times 100$, док колона $\#b$ приказује број инстанци за које је добијено решење једнако најбољем.

На основу резултата из Табеле 2.1, могу се извући следећи закључци:

- Метод ILP може оптимално да реши свих 45 инстанци.
- VNS такође долази до оптималних решења на свим инстанцама, а његова ефикасност се огледа у малом просечном времену извршавања (највеће време је $19.3s$).

Табела 2.1: Валидација оптималности VNS-а

			ILP		VNS			
инстанце	#	$\overline{best}$	$\overline{obj}$	$\#opt$	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	$\#b$
`MA`-20	15	45.5	45.5	15	1.6	45.5	0	15
`MA`-50	15	88.7	88.7	15	5.2	88.7	0	15
`MA`-100	15	151.8	151.8	15	19.3	151.8	0	15
All	45	95.3	95.3	45	8.7	95.3	0	45

Табела 2.2 даје упоредни приказ резултата метода ILP, VNS, GRASP и хибридног приступа GRASP+GA. Прве три колоне су идентичне онима у Табели 2.1, док наредна четири блока представљају резултате за сваки од метода. У блоку за ILP, колоне имају исто значење као у претходној табели, док су структуре осталих блокова аналогне блоку VNS из Табеле 2.1.

Анализом резултата из табеле 2.2 може се закључити следеће:

- ILP оптимално решава свих 45 инстанци у групи `AMS`-75, а исте резултате постиже и VNS. Друге две методе нису толико успешне - GRASP+GA проналази оптимална решења на 39 инстанци, док GRASP на само 17.

- На скуповима AMS-100 и AMS-125, ILP још увек успева да успешно реши велики број инстанци (70 од 90). Међутим, VNS проналази најбоље или резултате једнаке најбољим на 87 од 90 инстанци. Други најуспешнији метод је ILP, а затим следи GRASP+GA, који даје најбоље решење у 66 случајева. Најслабији је, очекивано, метод GRASP. Слични закључци могу се извести и анализом просечног квалитета решења.

- За инстанце из референтног скупа NEW (где је $|V| \in \{250, 500, 1000\}$), метод ILP се показао као неприменљив јер није могао да реши ниједну од 135 инстанци. Са друге стране, метод VNS даје боље резултате од осталих хеуристичких приступа. Ово је посебно изражено код највеће групе (45 инстанци са $|V| = 1000$), где је просечан квалитет решења добијених помоћу VNS 3076.8, док је други најбољи метод GRASP+GA са 3193.5. За 107 инстанци, VNS постиже најбоље резултате (или једнаке најбољим), док је GRASP+GA други најуспешнији са 64 најбоља решења. Као што је очекивано, просечно време извршавања два најуспешнија приступа је упоредиво.

Табела 2.2: VNS у односу на GRASP и GRASP+GA

			ILP		VNS				GRASP				GRASP+GA			
инстанце	#	$\overline{best}$	$\overline{obj}$	#*opt*	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	#*b*	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	#*b*	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	#*b*
AMS-75	45	421.7	**421.7**	45	11.5	**421.7**	0	45	1.3	443.8	4	17	9.9	423.9	0.34	39
AMS-100	45	524.5	**524.5**	44	20.2	524.6	0.01	44	2.5	539.2	2.3	16	23.1	526.5	0.32	32
AMS-125	45	610.1	**610.1**	26	31.1	610.4	0.03	43	5	636.9	3.49	12	47.8	611.8	0.23	34
NEW-250	45	1030.9	1149.9	0	198.5	**1032.4**	0.1	40	25.8	1115.3	7.41	1	196.6	1035.1	0.35	29
NEW-500	45	1763.5	2408.4	0	1068	**1770.5**	0.29	33	240.5	1908.3	7.42	0	1136.8	1773.9	0.54	24
NEW-1000	45	3059.7	5932.5	0	1813.3	**3076.8**	0.4	34	1482.2	3319.3	7.75	0	1834.2	3193.5	4.2	11
All	270	1235.1	1841.2	115	523.8	**1239.4**	0.14	239	292.9	1327.1	5.4	46	541.4	1260.8	1	169

2.4.4 Детаљни резултати

Имајући у виду да су сумарни резултати показали да VNS постиже оптимална решења на готово свим инстанцама са до 100 чворова (134 од 135 оваквих инстанци је оптимално решено), у наставку неће бити даље анализиране. Пажња ће бити усмерена на инстанце средње величине из скупа AMS-125 и велике инстанце из скупа NEW.

Табела 2.3 даје детаљно поређење методе VNS са методама ILP, GRASP и GRASP+GA на AMS-125 инстанцама. Сваки ред у табели представља резултате за једну инстанцу. Прва колона садржи ознаку инстанце у формату AMS-$|V|$-*p*-*c*-*w*-*id*, где параметри $(|V|, p, c, w)$ описују карактеристике инстанце (за више информација о генерисању инстанци погледати рад [6]). Следећа колона приказује најбољи познат резултат за дату инстанцу, након чега следе четири блока резултата.

Први блок представља резултате метода ILP, где колона *obj* приказује квалитет решења, а колона *ind.* статус завршетка алгоритма. Ознака *TL* (енг. *time limit*) означава да је достигнуто временско ограничење, што значи да оптималност решења није потврђена, док *opt* указује на оптимално решење. Наредна три блока приказују резултате метода VNS, GRASP и GRASP+GA. У сваком блоку налазе се четири колоне: време извршавања у секундама (t), квалитет решења (obj), процентуална разлика у односу на најбоље познато решење за дату инстанцу (pg) и индикаторска колона ($ind.$). Индикаторска колона има три могуће вредности: *best* означава да је постигнуто најбоље познато решење, *opt* означава да је решење и оптимално, док празно поље означава да није у питању ниједна од претходне две опције.

На основу детаљних резултата приказаних у табели 2.3, могу се извући следећи закључци:

- На свим инстанцама, изузев две где је $p = 0.2$ (релативно ретки графови), VNS постиже најбоље познато решење. Са друге стране, GRASP+GA не долази до најбољег решења на 11 инстанци.
- Највећа процентуална разлика у односу на најбоље решење за методу VNS износи 1.33%.
- VNS успева да пронађе оптимално решење у 25 од 26 случајева у којима је ILP дао оптимална решења, док GRASP+GA ово постиже у 19 случајева.

Табела 2.3: Детаљно поређење VNS-а у односу на GRASP и GRASP+GA на скупу AMS-125

		ILP		VNS				GRASP				GRASP+GA			
инстанца	*best*	*obj*	*ind.*	*t*	*obj*	*pg%*	*ind.*	*t*	*obj*	*pg%*	*ind.*	*t*	*obj*	*pg%*	*ind.*
AMS-125-0.2-10-50-1	1026	**1026**	TL	28.8	**1026**	0	best	2	1112	8.38		24	**1026**	0	best
AMS-125-0.2-10-50-2	1038	**1038**	TL	28.1	**1038**	0	best	2	1069	2.99		22	**1038**	0	best
AMS-125-0.2-10-50-3	935	**935**	opt	24.5	**935**	0	opt	2	1124	20.21		23	947	1.28	
AMS-125-0.2-10-50-4	1050	**1050**	TL	26.1	1064	1.33		2	1121	6.76		21	1051	0.1	
AMS-125-0.2-10-50-5	974	**974**	TL	25.7	**974**	0	best	2	1112	14.17		25	975	0.1	
AMS-125-0.2-25-25-1	720	**720**	opt	25.7	**720**	0	opt	2	803	11.53		26	**720**	0	opt
AMS-125-0.2-25-25-2	746	**746**	opt	22.8	**746**	0	opt	2	768	2.95		24	748	0.27	
AMS-125-0.2-25-25-3	715	**715**	opt	26.9	**715**	0	opt	2	752	5.17		21	717	0.28	
AMS-125-0.2-25-25-4	701	**701**	opt	25.4	**701**	0	opt	2	726	3.57		22	705	0.57	
AMS-125-0.2-25-25-5	684	**684**	opt	23.4	685	0.15		2	747	9.21		23	697	1.9	
AMS-125-0.2-50-10-1	455	**455**	opt	23.2	**455**	0	opt	2	457	0.44		21	**455**	0	opt
AMS-125-0.2-50-10-2	477	**477**	opt	22.4	**477**	0	opt	2	493	3.35		23	**477**	0	opt
AMS-125-0.2-50-10-3	490	**490**	opt	22	**490**	0	opt	2	501	2.24		21	**490**	0	opt
AMS-125-0.2-50-10-4	467	**467**	opt	22.5	**467**	0	opt	2	504	7.92		23	**467**	0	opt
AMS-125-0.2-50-10-5	457	**457**	opt	23.1	**457**	0	opt	2	468	2.41		24	459	0.44	
AMS-125-0.5-10-50-1	817	**817**	TL	33.8	**817**	0	best	4	**817**	0	best	41	**817**	0	best
AMS-125-0.5-10-50-2	815	**815**	TL	34.4	**815**	0	best	5	827	1.47		45	**815**	0	best
AMS-125-0.5-10-50-3	836	**836**	TL	35.8	**836**	0	best	4	880	5.26		45	872	4.31	
AMS-125-0.5-10-50-4	867	**867**	TL	33.7	**867**	0	best	4	914	5.42		55	**867**	0	best
AMS-125-0.5-10-50-5	867	**867**	TL	28.6	**867**	0	best	5	906	4.5		55	**867**	0	best
AMS-125-0.5-25-25-1	566	**566**	TL	32	**566**	0	best	5	**566**	0	best	48	**566**	0	best
AMS-125-0.5-25-25-2	533	**533**	opt	28.6	**533**	0	opt	5	561	5.25		48	**533**	0	opt
AMS-125-0.5-25-25-3	538	**538**	opt	30.4	**538**	0	opt	5	567	5.39		49	**538**	0	opt
AMS-125-0.5-25-25-4	552	**552**	TL	31.1	**552**	0	best	4	565	2.36		53	**552**	0	best
AMS-125-0.5-25-25-5	545	**545**	TL	31.7	**545**	0	best	5	548	0.55		48	548	0.55	
AMS-125-0.5-50-10-1	334	**334**	opt	29.3	**334**	0	opt	4	336	0.6		40	**334**	0	opt
AMS-125-0.5-50-10-2	330	**330**	opt	28.5	**330**	0	opt	4	**330**	0	opt	38	**330**	0	opt
AMS-125-0.5-50-10-3	315	**315**	opt	26.5	**315**	0	opt	5	**315**	0	opt	49	**315**	0	opt
AMS-125-0.5-50-10-4	316	**316**	opt	29.2	**316**	0	opt	5	**316**	0	opt	51	**316**	0	opt
AMS-125-0.5-50-10-5	311	**311**	opt	28.9	**311**	0	opt	4	**311**	0	opt	40	**311**	0	opt
AMS-125-0.8-10-50-1	793	**793**	TL	39.5	**793**	0	best	9	**793**	0	best	78	**793**	0	best
AMS-125-0.8-10-50-2	845	**845**	TL	40.4	**845**	0	best	8	854	1.07		72	**845**	0	best
AMS-125-0.8-10-50-3	787	**787**	TL	41.8	**787**	0	best	9	829	5.34		74	**787**	0	best
AMS-125-0.8-10-50-4	777	**777**	TL	40.8	**777**	0	best	9	829	6.69		83	**777**	0	best
AMS-125-0.8-10-50-5	813	**813**	TL	40.7	**813**	0	best	8	827	1.72		77	**813**	0	best
AMS-125-0.8-25-25-1	508	**508**	opt	37.4	**508**	0	opt	9	521	2.56		69	510	0.39	
AMS-125-0.8-25-25-2	498	**498**	opt	38.3	**498**	0	opt	9	499	0.2		65	**498**	0	opt
AMS-125-0.8-25-25-3	513	**513**	TL	37	**513**	0	best	9	523	1.95		77	**513**	0	best
AMS-125-0.8-25-25-4	493	**493**	opt	38.9	**493**	0	opt	8	506	2.64		75	**493**	0	opt
AMS-125-0.8-25-25-5	504	**504**	TL	38.3	**504**	0	best	8	519	2.98		76	**504**	0	best
AMS-125-0.8-50-10-1	307	**307**	opt	33.6	**307**	0	opt	8	**307**	0	opt	64	**307**	0	opt
AMS-125-0.8-50-10-2	296	**296**	opt	37.6	**296**	0	opt	8	**296**	0	opt	57	**296**	0	opt
AMS-125-0.8-50-10-3	294	**294**	opt	33.6	**294**	0	opt	8	**294**	0	opt	71	**294**	0	opt
AMS-125-0.8-50-10-4	270	**270**	opt	34.8	**270**	0	opt	9	**270**	0	opt	86	**270**	0	opt
AMS-125-0.8-50-10-5	278	**278**	opt	33.6	**278**	0	opt	9	**278**	0	opt	77	**278**	0	opt

Резултати за инстанце из скупа NEW са 250 и 500 чворова дати су у додатку Б, у табелама

Б.1 и Б.2. Табела 2.4 приказује резултате за NEW инстанце са 1000 чворова. Из ње се могу извести следећи закључци:

- ILP успева да пронађе прималне границе само за ретке графове ($p = 0.2$). У другим случајевима, модел није могао бити изграђен због ограничења меморије (6GB), што је означено као *ML* (енг. *memory limit*).
- VNS проналази најбоља решења за 34 од 45 инстанци, док је GRASP+GA то успева у само 11 случајева.
- Интересантно је да код ретких графова ($p = 0.2$), GRASP+GA углавном даје боље резултате од VNS. Међутим, за гушће графове ($p \in 0.5, 0.8$), VNS постиже најбоље резултате у свим случајевима, док GRASP+GA не проналази ниједно најбоље решење.

Табела 2.4: Детаљно поређење VNS-а у односу на GRASP и GRASP+GA на скупу NEW-1000

		ILP		VNS				GRASP				GRASP+GA			
инстанца	$\overline{best}$	$\overline{obj}$	*ind.*	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\bar{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*
NEW-1000-0.2-10-50-1	4990	10851	TL	1802.9	5106	2.32		809.3	5661	13.45		1844.6	**4990**	0	best
NEW-1000-0.2-10-50-2	5233	9687	TL	1806.7	5316	1.59		824.3	5803	10.89		1912	**5233**	0	best
NEW-1000-0.2-10-50-3	5044	9995	TL	1800.1	**5044**	0	best	829	5790	14.79		2030.3	5196	3.01	
NEW-1000-0.2-10-50-4	5079	9760	TL	1803.5	5321	4.76		829.2	5742	13.05		1901.6	**5079**	0	best
NEW-1000-0.2-10-50-5	5098	10617	TL	1802.4	5260	3.18		811.6	5887	15.48		1817.7	**5098**	0	best
NEW-1000-0.2-25-25-1	3167	5340	TL	1803.2	3233	2.08		804.9	3525	11.3		1903.5	**3167**	0	best
NEW-1000-0.2-25-25-2	3163	5845	TL	1804.6	**3163**	0	best	827.8	3611	14.16		1954.9	3179	0.51	
NEW-1000-0.2-25-25-3	3148	5003	TL	1800.7	**3148**	0	best	810.7	3520	11.82		1802.9	3159	0.35	
NEW-1000-0.2-25-25-4	3227	4863	TL	1802.4	3268	1.27		808.1	3583	11.03		1839.1	**3227**	0	best
NEW-1000-0.2-25-25-5	3234	5349	TL	1801	**3234**	0	best	813.6	3600	11.32		1848.6	3285	1.58	
NEW-1000-0.2-50-10-1	1907	2419	TL	1807.6	1930	1.21		802.7	2072	8.65		1876.6	**1907**	0	best
NEW-1000-0.2-50-10-2	1882	2275	TL	1803.2	1890	0.43		812.5	2034	8.08		1826.3	**1882**	0	best
NEW-1000-0.2-50-10-3	1899	2401	TL	1820.8	1921	1.16		802.3	2042	7.53		1851.7	**1899**	0	best
NEW-1000-0.2-50-10-4	1914	2281	TL	1801.6	1916	0.1		816.2	2133	11.44		1842	**1914**	0	best
NEW-1000-0.2-50-10-5	1934	2302	TL	1807.3	1936	0.1		807.5	2095	8.32		1883.9	**1934**	0	best
NEW-1000-0.5-10-50-1	4483	-	ML	1817.9	**4483**	0	best	1813.7	4918	9.7		1817.8	4869	8.61	
NEW-1000-0.5-10-50-2	4462	-	ML	1802.6	**4462**	0	best	1805.5	4857	8.85		1814.3	4807	7.73	
NEW-1000-0.5-10-50-3	4567	-	ML	1809.1	**4567**	0	best	1820.4	4983	9.11		1819.8	5015	9.81	
NEW-1000-0.5-10-50-4	4480	-	ML	1847.1	**4480**	0	best	1811.4	4855	8.37		1804.8	4967	10.87	
NEW-1000-0.5-10-50-5	4469	-	ML	1812.2	**4469**	0	best	1822.7	4765	6.62		1805.9	4862	8.79	
NEW-1000-0.5-25-25-1	2743	-	ML	1826.7	**2743**	0	best	1821.6	2927	6.71		1822.6	2961	7.95	
NEW-1000-0.5-25-25-2	2771	-	ML	1814.8	**2771**	0	best	1817.7	2914	5.16		1811.8	2905	4.84	
NEW-1000-0.5-25-25-3	2783	-	ML	1813.2	**2783**	0	best	1820.9	2940	5.64		1815.7	2923	5.03	
NEW-1000-0.5-25-25-4	2757	-	ML	1800	**2757**	0	best	1812.2	2987	8.34		1812	2942	6.71	
NEW-1000-0.5-25-25-5	2752	-	ML	1800.1	**2752**	0	best	1822	2949	7.16		1818.3	2946	7.05	
NEW-1000-0.5-50-10-1	1612	-	ML	1801.3	**1612**	0	best	1801.7	1702	5.58		1810.5	1699	5.4	
NEW-1000-0.5-50-10-2	1631	-	ML	1808	**1631**	0	best	1816.3	1698	4.11		1814	1697	4.05	
NEW-1000-0.5-50-10-3	1625	-	ML	1801.3	**1625**	0	best	1804.3	1699	4.55		1819.2	1711	5.29	
NEW-1000-0.5-50-10-4	1629	-	ML	1836	**1629**	0	best	1821.5	1714	5.22		1805	1710	4.97	
NEW-1000-0.5-50-10-5	1597	-	ML	1825.2	**1597**	0	best	1814.1	1677	5.01		1814	1694	6.07	
NEW-1000-0.8-10-50-1	4334	-	ML	1806.3	**4334**	0	best	1803.1	4518	4.25		1827.1	4678	7.94	
NEW-1000-0.8-10-50-2	4352	-	ML	1800.3	**4352**	0	best	1813.7	4577	5.17		1807.2	4630	6.39	
NEW-1000-0.8-10-50-3	4358	-	ML	1841.2	**4358**	0	best	1821.3	4623	6.08		1800.8	4698	7.8	
NEW-1000-0.8-10-50-4	4375	-	ML	1818.5	**4375**	0	best	1829.3	4713	7.73		1817.2	4660	6.51	
NEW-1000-0.8-10-50-5	4373	-	ML	1857.4	**4373**	0	best	1816.4	4606	5.33		1814	4587	4.89	
NEW-1000-0.8-25-25-1	2580	-	ML	1885	**2580**	0	best	1828.7	2719	5.39		1802.1	2723	5.54	
NEW-1000-0.8-25-25-2	2605	-	ML	1848.6	**2605**	0	best	1807.4	2756	5.8		1811.4	2736	5.03	
NEW-1000-0.8-25-25-3	2577	-	ML	1818.6	**2577**	0	best	1817.6	2752	6.79		1824.2	2742	6.4	
NEW-1000-0.8-25-25-4	2574	-	ML	1815.1	**2574**	0	best	1818.7	2647	2.84		1803.4	2668	3.65	
NEW-1000-0.8-25-25-5	2613	-	ML	1800.5	**2613**	0	best	1830.2	2773	6.12		1840.6	2726	4.32	
NEW-1000-0.8-50-10-1	1525	-	ML	1803.7	**1525**	0	best	1801.2	1594	4.52		1820.5	1597	4.72	
NEW-1000-0.8-50-10-2	1549	-	ML	1800.9	**1549**	0	best	1803.2	1632	5.36		1819.9	1637	5.68	
NEW-1000-0.8-50-10-3	1526	-	ML	1815.5	**1526**	0	best	1826.1	1624	6.42		1800.5	1594	4.46	
NEW-1000-0.8-50-10-4	1526	-	ML	1801	**1526**	0	best	1821	1566	2.62		1801.1	1579	3.47	
NEW-1000-0.8-50-10-5	1541	-	ML	1801.8	**1541**	0	best	1825.7	1584	2.79		1806.2	1596	3.57	

2.4.5 Статистичка анализа

Ради провере статистичке значајности уочених разлика између четири разматране методе, примењена је следећа статистичка методологија. Прво је примењен Фридманов тест [47] за анализу свих метода, прво засебно на скуповима `AMS` и `NEW`, а потом и на свим инстанцама из оба скупа. Након тога, у случају када је нулта хипотеза одбачена, где H_0 означава да су методе статистички једнаке, примењен је Неменјијев пост-хок тест [93] за упоредно поређење парова.

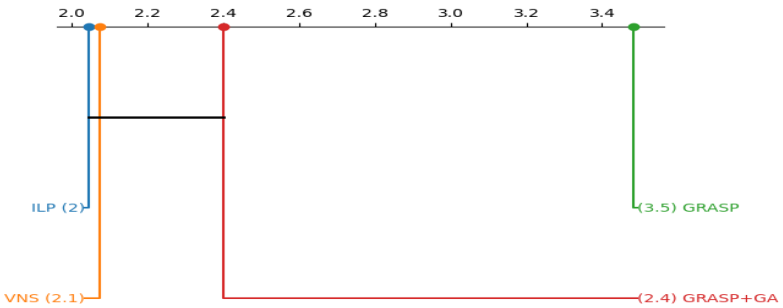
Резултати су приказани помоћу графика критичне разлике (енг. *critical difference*, CD). На овом графику свака метода је постављена на хоризонтално осу на позицији која одговара њеном просечном рангу на посматраном подскупу инстанци. Критична разлика је израчуната за ниво значајности $p = 0.05$. Уколико је разлика између метода мала, односно уколико није детектована статистички значајна разлика, каже се да су методе статистички једнаке и на графику се повезују хоризонталном линијом.

Резултати су подељени у три групе: прво су приказани резултати за све инстанце из скупа `AMS`, затим за све инстанце из скупа `NEW`, и на крају за све инстанце за оба скупа заједно. Одговарајући графици су приказани на сликама 2.3, 2.4 и 2.5 и из њих произлазе следећи закључци:

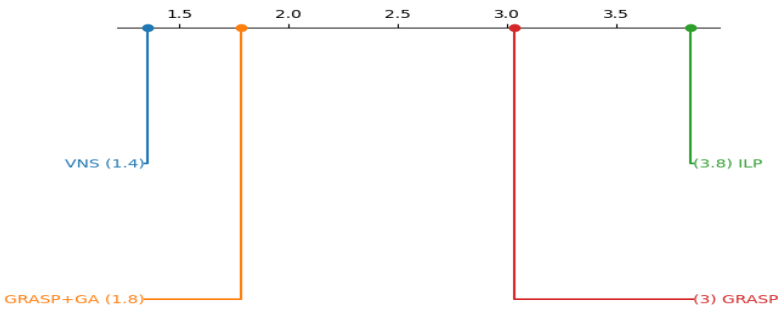
- На референтном скупу `AMS`, који се састоји од малих и средње великих инстанци, ILP успева да пронађе оптимална решења на 115 од укупно 135 инстанци. Стога не чуди што ова метода има најмањи просечан ранг (2.04) на овом скупу инстанци. VNS има незнатно лошији просечан ранг (2.07), док су преостале методе значајно лошије. Нема статистички значајне разлике између метода ILP, VNS и GRASP+GA, упркос наизглед лошијем просечном рангу GRASP+GA (2.40). Метода GRASP показује статистички значајно лошије резултате на овом скупу инстанци, са просечним рангом 3.48.

- На референтном скупу `NEW`, који се састоји од великих инстанци, постоји статистички значајна разлика између свих метода. Конкретно, VNS надмашује GRASP+GA, који је пак супериорнији у односу на GRASP. Као што је очекивано, ILP постиже најслабије резултате због немогућности решавања проблема већих димензија.

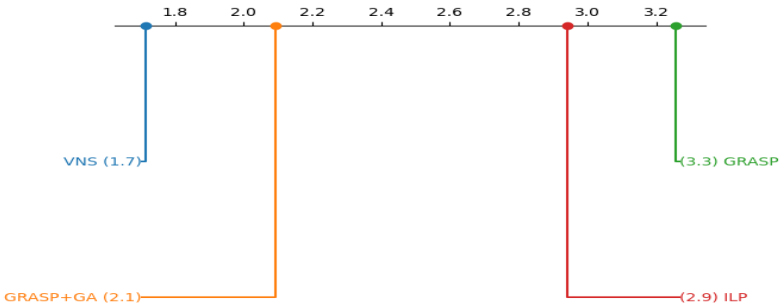
- Када се анализирају резултати на инстанцама из оба скупа заједно, уочава се да је VNS најбоља метода. Друга најбоља је GRASP+GA, затим следи ILP, а најслабија метода је GRASP.



Слика 2.3: График критичне разлике за резултате на свим инстанцама из референтног скупа `AMS`



Слика 2.4: График критичне разлике за резултате на свим инстанцама из референтног скупа NEW



Слика 2.5: График критичне разлике за резултате на свим инстанцама из референтних скупова AMS и NEW заједно

2.5 Примена у ширењу информација на друштвеним мрежама

У овом делу рада биће анализиран утицај избора почетних чворова (ширитеља) на процес ширења информација у великим друштвеним мрежама. Ширење информација, као што су гласине или вирални маркетинг, представља важан аспект комуникације и може утицати на целокупно друштво. Наиме, ширење информација може имати дубоке социјалне и економске последице, јер утиче на поверење у друштву, креирање јавног мњења, као и на маркетиншке стратегије компанија које зависе од брзог и ефикасног преношења информација. Због тога је разумевање како се информације крећу од једног до другог појединца, као и који фактори утичу на овај процес, кључно за ефикасно управљање информацијама у различитим контекстима, од политике и новинарства до комерцијалних кампања. Самим тим, не изненађује што је ова тема широко истраживана у области науке о подацима.

У неким од бројних радова на ову тему, попут [90, 116, 7], предложени су различити модели засновани на SIR парадигми. SIR модел се најчешће користи у епидемиологији и представља модел ширења заразних болести. У овом моделу, читава популација је подељена у три групе: S - подложни инфекцији (енг. *susceptible*), I - заразни (енг. *infected*) и R - опорављени (енг. *recovered*). Другим речима, S чворови су они који могу бити заражени, I чворови су они који су већ заражени и могу заразити S чворове, а R чворови су они који су се опоравили и не могу више ни да се инфицирају, нити да заразе друге чворове. У контексту ширења информација, терминологија је следећа:

- S чворови представљају оне који нису упознати са информацијом, називају се и неупућени (енг. *ignorants*);

- I чворови су они који су већ чули информацију и могу да је пренесу другим чворовима, односно ширитељи (енг. *spreaders*);

- R чворови су они који су чули информацију, али је не шире даље, називају се и гушитељи (енг. *stiflers*).

Основна правила ширења информација у оквиру SIR модела су следећа:

1. Када ширитељ ступи у контакт са неупућеним, овај постаје ширитељ са вероватноћом β (стопа ширења или инфективност).

2. Када ширитељ ступи у контакт са другим ширитељем или гушитељем, он постаје гушитељ са вероватноћом γ (стопа опоравка или имунитета).

Процес ширења почиње тако што се одређени подскуп чворова у мрежи означи као иницијално информисан, а завршава се када у графу више не постоје ширитељи.

Користећи овај модел, у раду [82] је анализирано ширење гласина на микроблог мрежама. У раду [80] је такође коришћен модел заснован на SIR парадигми, али је уместо гласина анализирано ширење информација у динамичким друштвеним мрежама. У раду [73] аутори користе SIR модел за симулацију ширења лажних вести путем апликације WhatsApp. Једна од комерцијално привлачних примена овог концепта је вирални маркетинг, где корисници друштвене мреже путем међусобних препорука могу значајно утицати на продају и препознатљивост производа или услуге. Рана истраживања у овој области дата су у [79], где се поред осталих модела разматра и SIR. Новија истраживања, попут рада [103], анализирају утицај параметара SIR модела на примену у маркетиншким кампањама.

У овом раду, фокус је на избору почетних чворова ширитеља. Размотримо случај у коме је скуп иницијалних ширитеља постављен на решење проблема MWTDS, при чему су параметри $\beta = 0$ и $\gamma = 1$. Сви остали чворови су неупућени, као и увек на почетку процеса. Према дефиницији MWTDS решења, сваки неупућени чвор у графу има суседа који је ширитељ. Међутим, пошто је $\beta = 0$, ниједан неупућени неће променити стање. Такође, сваки ширитељ има бар једног суседа који је такође ширитељ, па због $\gamma = 1$ сви ширитељи постају гушитељи након само једне итерације и, како више нема ширитеља, читав процес се завршава.

Када параметри β и γ нису постављени на вредности 0 и 1, процес ширења информација више није детерминистички. У сваком случају, хипотеза је да ће избор почетних ширитеља на основу решења проблема MWTDS довести до бржег ширења информација у односу на насумично одабран скуп ширитеља исте кардиналности.

Извршени су експерименти на SNAP скупу података са друштвеним мрежама [78]. Инстанце из овог скупа су реални, велики графови који представљају делове популарних друштвених мрежа попут Фејсбука и Твитера, који су од посебног значаја у области ширења информација. Коришћено је 27 различитих инстанци, величине од 1912 до 81306 чворова и од 31299 до 1342310 грана.

У табели 2.5 приказани су резултати извршавања на свим инстанцама, а последњи ред садржи просечне резултате. Метрика која је коришћена је време конвергенције, односно број итерација до завршетка ширења. Прве три колоне приказују назив инстанце, број чворова ($|V|$) и број грана ($|E|$). Тестиране су различите комбинације параметара $(\beta, \gamma) \in \{0.1, 0.5, 0.9\} \times \{0.1, 0.5, 0.9\}$. За сваку комбинацију приказане су две колоне: у првој је време конвергенције када се користи MWTDS решење добијено методом Vns (означено као V.), а у другој када се користи насумичан избор иницијалних ширитеља (означено као R.).

Табела 2.5: Поређење времена конвергенције

			(0.1, 0.1)		(0.1, 0.5)		(0.1, 0.9)		(0.5, 0.1)		(0.5, 0.5)		(0.5, 0.9)		(0.9, 0.1)		(0.9, 0.5)		(0.9, 0.9)	
инстанца	$\|V\|$	$\|E\|$	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.	V.	R.
deezer_HR	54573	498202	80	75	10	64	10	49	77	90	11	16	7	10	86	69	14	15	6	7
deezer_HU	47538	222887	71	82	13	70	12	69	63	76	15	13	7	8	87	75	16	16	6	8
deezer_RO	41773	125826	91	93	15	67	10	60	91	71	17	17	8	11	94	80	16	18	6	8
Slashdot0811	77360	546487	48	48	12	10	6	8	48	57	9	11	6	8	46	51	11	10	5	9
Slashdot0902	82168	582533	63	67	16	84	6	50	65	80	12	13	6	9	75	84	15	14	6	7
Wiki-Vote	7115	100762	61	52	12	43	5	30	87	78	11	14	6	8	106	80	14	18	6	6
facebook_artist	50515	819306	58	88	15	43	8	51	72	71	14	20	6	8	75	126	15	16	6	8
facebook_athletes	13866	86858	60	85	12	53	9	29	55	70	12	16	6	8	73	61	14	16	6	7
facebook_company	14113	52310	75	72	14	48	9	48	76	61	11	16	6	15	96	77	13	12	6	8
deezer_europe	28281	92752	71	80	15	53	9	62	104	82	14	16	8	9	81	95	17	15	7	10
facebook_combined	4039	88234	38	28	9	14	11	18	45	96	9	11	6	9	39	39	9	12	4	8
facebook_government	7057	89455	49	47	12	29	11	36	67	41	11	13	6	9	48	61	11	12	6	7
lastfm_asia	7624	27806	67	67	11	54	7	85	94	81	13	19	6	8	68	63	11	14	6	8
musae_DE	9498	153138	37	62	8	41	7	19	66	57	10	14	6	6	52	76	12	13	5	7
musae_ENGB	7126	35324	67	52	10	24	8	40	65	69	11	16	5	7	99	84	12	15	5	8
musae_ES	4648	59382	37	50	9	29	8	41	68	40	10	10	6	7	50	53	10	12	5	9
musae_FR	6549	112666	53	38	8	18	8	7	54	80	13	10	5	6	65	96	12	10	6	7
musae_PTBR	1912	31299	21	43	13	11	6	7	50	61	8	11	5	6	60	56	8	10	5	7
musae_RU	4385	37304	72	42	10	38	7	36	60	59	10	13	5	9	57	65	14	12	5	6
musae_facebook	22470	171002	64	92	12	56	9	47	88	78	11	16	7	9	74	84	12	14	6	9
musae_git	37700	289003	73	71	13	64	7	44	95	90	14	16	6	8	115	85	16	15	7	7
facebook_new_sites	27917	206259	55	60	14	65	9	45	93	53	16	15	8	9	76	106	12	13	6	8
facebook_politician	5908	41729	50	82	12	44	9	47	53	84	12	13	6	11	53	61	11	13	5	9
facebook_public_figure	11565	67114	53	103	15	65	8	60	74	70	15	16	6	12	71	87	15	15	5	9
soc-Epinions1	75879	405740	94	107	14	80	7	139	107	102	19	19	7	15	100	104	18	21	7	10
facebook_tvshow	3892	17262	58	71	10	36	9	45	60	55	12	19	6	12	62	67	12	14	6	7
twitter_combined	81306	1342310	56	53	11	15	13	20	74	116	13	21	7	8	82	80	15	15	6	8
Average	27288	233442.6	60.1	67	12	45.1	8.4	44.1	72.3	72.9	12.3	15	6.3	9.1	73.7	76.5	13.1	14.1	5.7	7.9

Из добијених резултата могу се издвојити следећи закључци:

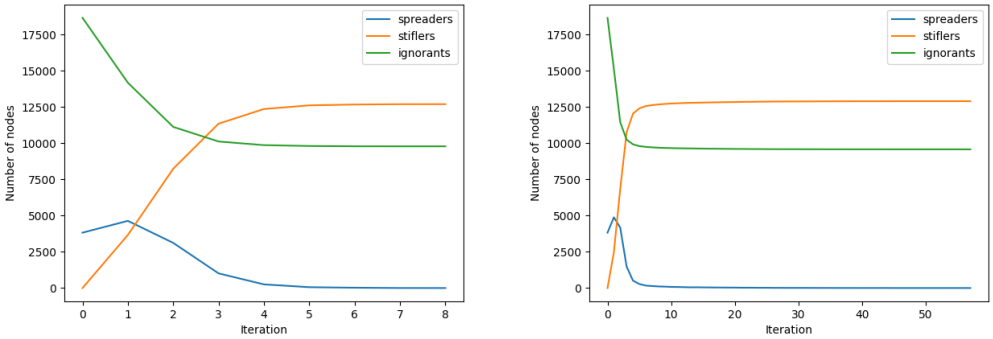
- Када је стопа ширења ниска, тј. $\beta = 0.1$, MWTDS приступ обезбеђује знатно бржу конвергенцију у односу на насумичан избор. Посебно се истиче случај ($\beta = 0.1, \gamma = 0.9$), где је просечно време конвергенције 8.4 за MWTDS у поређењу са 44.1 за насумичан избор. Овакви резултати су очекивани, с обзиром на то да када је стопа ширења ниска, избор иницијалних ширитеља има већи значај.

- Насупрот томе, при веома високој стопи ширења (што је у пракси ређе), квалитет избора почетних ширитеља постаје мање битан. И у таквим случајевима MWTDS решење показује боље резултате од насумичног избора, али је разлика мања. На пример, у случају ($\beta = 0.9, \gamma = 0.5$), време конвергенције је 13.1 (MWTDS) наспрам 14.1 (насумично).

- Више стопе опоравка γ такође погодују примени MWTDS решења, јер отежавају ширење информације. То се може емпиријски показати праћењем релативне ефикасности при фиксном β и растућем γ. На пример, за $\beta = 0.9$, однос просечног времена конвергенције (просек колоне V. подељен просеком колоне R.) се смањује: 0.96, 0.93, 0.72.

- Закључак је да комбинација ниске стопе ширења и високе стопе опоравка представља најповољнији сценарио за примену Vns MWTDS решења.

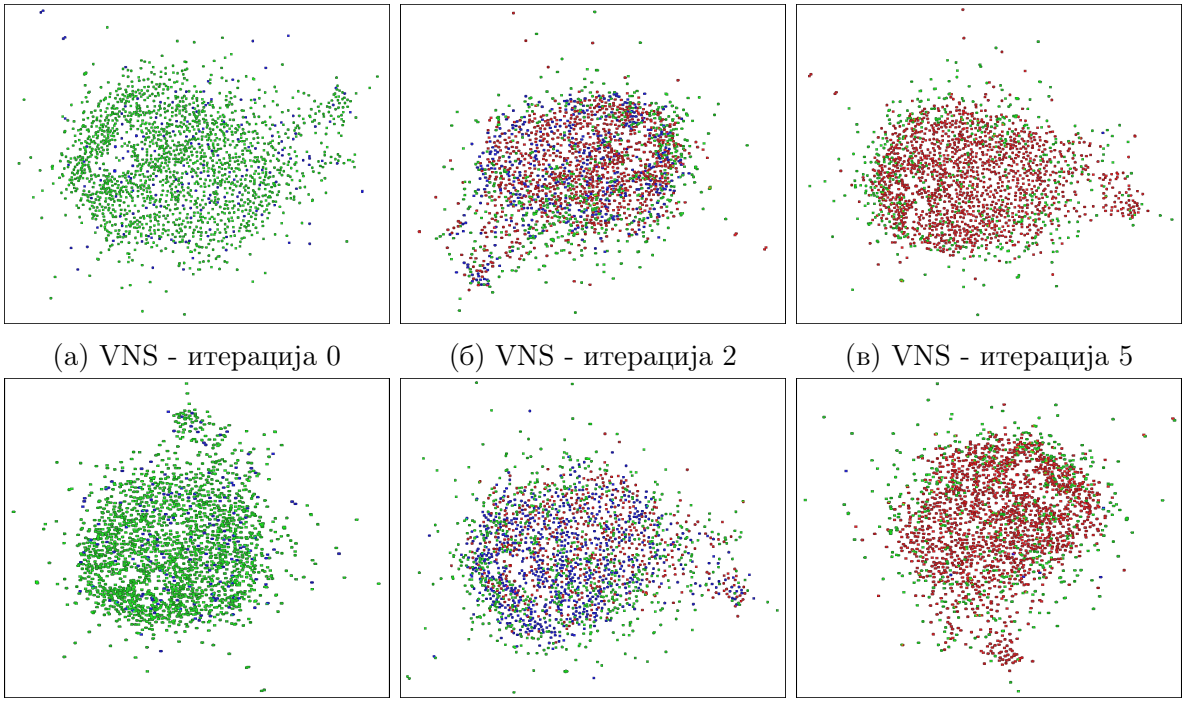
Треба нагласити да је коначна расподела ширитеља, неупућених и гушитеља, након завршетка процеса слична, као што је приказано на слици 2.6. Дакле, насумичним избором почетних ширитеља може се доћи до прилично добре покривености мреже. Међутим, у том случају је потребно много више итерација да би се постигла конвергенција, што је главна предност MWTDS решења добијеног применом предложеног VNS алгоритма.



(а) VNS (б) Насумично

Слика 2.6: SIR динамика на инстанци musae-facebook са $\beta = 0.1$ and $\gamma = 0.9$. Иницијални скуп ширитеља је:
(а) MWTDS решење добијеном употребом VNS-а,
(b) насумичан скуп чворова исте кардиналности као у (а).

Упоредимо сада расподеле ширитеља, неупућених и гушитеља по итерацијама, што је на другачији начин приказано на слици 2.7. На самом почетку, број ширитеља, неупућених и гушитеља је исти у оба случаја. Већ у другој итерацији примећује се јасна предност MWTDS решења: број неупућених у MWTDS случају је 913, у поређењу са 488 у случају насумичне иницијализације. У петој итерацији MWTDS приступа остаје само један ширитељ, а процес се завршава већ у шестој. Насупрот томе, насумични приступ траје још три итерације и завршава се тек у осмој.



(а) VNS - итерација 0 (б) VNS - итерација 2 (в) VNS - итерација 5

(г) Насумично - итерација 0 (д) Насумично - итерација 2 (ђ) Насумично - итерација 5

Слика 2.7: SIR динамика кроз итерације на инстанци musae-PTBR са $\beta = 0.1$ and $\gamma = 0.9$. Ширитељи су приказани плавом, гушитељи црвеном, а неупућени зеленом бојом.

2.6 Завршна разматрања

У овом поглављу развијена је ефикасна метода променљивих околина за проблем минималне тежинске тоталне доминације, заснована на размрдавању и две варијанте локалне претраге. Кључан допринос представља пажљиво дефинисана функција прилагођености и њено инкрементално израчунавање, што омогућава стабилно и ефикасно кретање кроз допустиве и недопустиве регионе простора претраге. Експериментални резултати показују да предложени алгоритам достиже оптимална решења на скоро свим малим и средњим инстанцама, док на великим графовима надмашује конкурентске приступе. Приказана примена на модел ширења информација у друштвеним мрежама додатно илуструје практичну релевантност проблема, као и развијене методе. Постигнути резултати постављају основу за даље истраживање и примену метахеуристичких приступа у домену великих комплексних мрежа.

Поглавље 3

Проблем минималне тежинске независне доминације

Независна доминација је варијација класичног проблема доминације, где постоји додатно ограничење да ниједна два чвора из решења не смеју бити суседи. Овај проблем је нашао примену у разним областима, као што су: одабир атрибута (енг. *feature selection*) [105, 11, 3], формирање виртуелне окоснице мреже (енг. *backbone network*) у ад-хок мрежама [114], те кластеровање у бежичним сензорским мрежама [111]. Природно се може проширити увођењем тежина чворова и грана како би се боље описали сложенији реални проблеми. Тако настаје проблем минималне тежинске независне доминације (MWIDS), уведен у раду [23], који је главна тема овог поглавља. Очекивано, и тежинска варијанта овог проблема применљива је у сличним сценаријима, јер омогућава додељивање тежина, како чворовима, тако и гранама графа. На пример, у контексту кластеровања бежичних мрежа, тежине чворова могу бити одређене различитим факторима, као што су степен чвора, покретљивост и преостала енергија [25], док тежине грана природно могу представљати растојања између чворова.

Садржај овог поглавља се заснива на резултатима објављеним у раду [67].

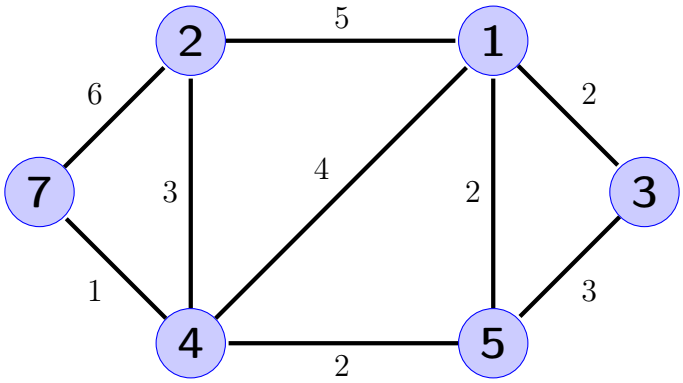
3.1 Дефиниција проблема

Скуп $S \subseteq V$ назива се независним ако за сваки пар чворова $u, v \in S$ не постоји грана која их повезује $\{u, v\} \in E$. Скуп S је независан доминирајући скуп ако је истовремено и независан и доминирајући. Проблем тежинске независне доминације разматра неусмерене графове са позитивним тежинама придруженим чворовима и гранама. Циљ је пронаћи независан доминирајући скуп S који минимизује следећу функцију циља:

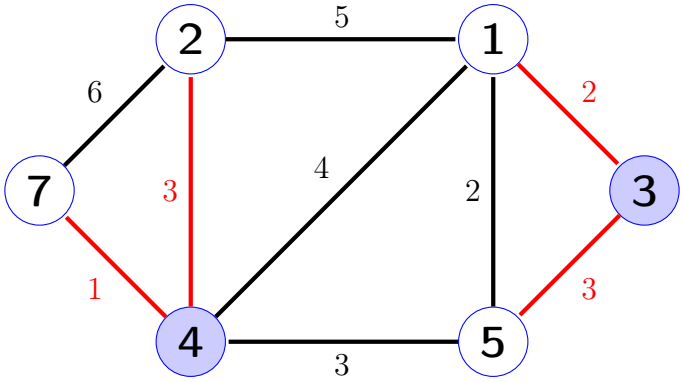
$$obj(S) = \sum_{v \in S} w_v + \sum_{v \in V \setminus S} \min\{w_e \mid e = \{v, u\}, u \in S\} \tag{3.1}$$

Вредност функције циља за скуп S добија се као збир тежина чворова који припадају скупу S, уз додатак тежина грана најмање тежине које повезују чворове ван скупа S са чворовима унутар скупа S.[1] Практична примена овог проблема проналази се у алгоритмима за кластеровање у мобилним бежичним ад-хок мрежама, где скуп доминирајућих чворова представља такозване управљаче кластера (енг. *cluster heads*). Оваква функција циља је мотивисана потребом да се истовремено минимизује цена успостављања тих центара (тежине чворова) и цена комуникације, односно повезивања преосталих корисника са њима (тежине грана).

[1] У поглављу 2 су овакве гране биле називане спољашње гране, а сада не постоје унутрашње, јер по дефиницији проблема нема грана између чворова унутар решења.



Слика 3.1: Пример инстанце проблема MWIDS



Слика 3.2: Оптимално решење проблема MWIDS

3.1.1 Пример инстанце

На слици 3.1 је приказан пример инстанце проблема MWIDS. Тежине чворова су означене бројевима унутар њих, а тежине грана на њима. Оптимално решење проблема MWIDS за ову инстанцу је приказано на слици 3.2. Решење се састоји од чворова обојених плавом бојом, а гране које су обојене црвеном бојом представљају гране минималне тежине које повезују чворове ван решења са чворовима из решења. Вредност функције циља, односно збир тежина плавих чворова и црвених грана са слике, за ово решење износи $(4+3)+(1+3+2+3)=7+9=16$.

3.2 Преглед литературе

У раду [23] предложен је алгоритам линеарне временске сложености за решавањее проблема MWIDS на серијски-паралелним графовима (енг. *series-parallel graphs*). За случај општих графова, у литератури постоје два рада. У раду [37] предложено је више приступа:

- Три модела целобројног линеарног програмирања:

  1. ILP-1 је модел који користи три скупа индикаторских променљивих: бинарне променљиве x_v које означавају да ли је чвор v изабран у решење, бинарне променљиве y_e које означавају да ли је грана e подобна за избор, и бинарне променљиве z_e које означавају да ли је грана e изабрана ради повезивања чвора ван решења са неким из решења. Укупно, модел има $|V|+2\cdot|E|$ променљивих и $3\cdot|V|+5\cdot|E|$ ограничења.

2. ILP-2 је сличан модел који елиминише скуп променљивих y_e. Ограничења везана за y_e су замењена новим скупом неједначина над x и z променљивама, па је број променљивих смањен на $|V| + |E|$, а укупан број ограничења: $3 \cdot |V| + 4 \cdot |E|$.
3. ILP-3 је структурно другачији модел, који се заснива на експлицитном моделовању доприноса тежина грана за сваки чвор, коришћењем целобројних променљивих q_v. Укупно користи $2 \cdot |V|$ променљивих и $5 \cdot |V| + 3 \cdot |E|$ ограничења.

- Два похлепна алгоритма:
  1. GREEDY-1 почиње од празног решења S и итеративно додаје чвор v који има највећу вредност $\frac{|N_{G'}(v)|}{w_v}$, где G' представља преостали граф након претходно изабраних чворова, а $N_{G'}(v)$ је скуп суседа чвора v у графу G'. Након додавања чвора v у S, тај чвор, његови суседи и све њихове инцидентне гране се уклањају из графа G'.
  2. GREEDY-2 додатно узима у обзир и тежине грана. Конкретно, користи се помоћна функција циља f^{aux} дефинисана као збир доприноса чворова у односу на парцијално решење S. За чвор $v \in S$ допринос је w_v, док је за чвор $v \notin S$ допринос дефинисан као $\min\{w_{uv} \mid u \in S\}$ ако има суседа у S, а у супротном се поставља на максималну тежину гране у целом графу.
- Популациони итеративни похлепни алгоритам PBIG (енг. *Population-Based Iterative Greedy*) – метахеуристика заснована на пробабилистичкој верзији GREEDY-2 хеуристике, која је примењена самостално и у комбинацији са Construct-Merge-Solve-Adapt (CMSA) алгоритмом [17].

Најбоље резултате постиже алгоритам локалне претраге инспирисан учењем поткрепљивањем предложен у раду [110]. У оквиру овог приступа дефинисане су три различите функције за бодовање (енг. *scoring functions*), прилагођене различитим својствима тежина чворова и грана, које се користе у оквиру похлепне хеуристике и алгоритма локалне претраге.

3.3 Предложени ILP модели за проблем MWIDS

У овом одељку биће представљена два ILP модела за решавање проблема MWIDS који, као што ће бити показано у секцији 3.5, надмашују резултате досадашњих модела из рада [37].

3.3.1 ILP модел NEW-1

Први од два нова предложена ILP модела, означен са NEW-1, користи два скупа бинарних индикаторских променљивих:

- x_u – бинарна променљива која означава да ли је чвор u изабран у решење,
- y_{uv} – бинарна променљива која означава да ли је чвор $v \in V$ повезан са чвором $u \in S$, односно да ли чвор u покрива чвор v.

У овом случају, потребно је разликовати гране (u, v) и (v, u), па је стога потребно увести скуп усмерених грана $A = \{(u, v) \cup (v, u) \mid \forall e = \{u, v\} \in E\}$, где за сваку неусмерену грану постоји пар усмерених. Пре саме формулације модела, неопходно је дефинисати два скупа:

- $\delta^-(u)$ – скуп свих усмерених грана које улазе у чвор u,
- $\delta^+(u)$ – скуп свих усмерених грана које излазе из чвора u.

Користећи ову нотацију, MWIDS проблем се може формулисати на следећи начин:

$$\min \sum_{u \in V} w_u x_u + \sum_{(u,v) \in A} w_{uv} y_{uv} \tag{3.2}$$

уз услове

$$x_u + x_v \leq 1, \quad \forall e = \{u, v\} \in E \tag{3.3}$$

$$x_u + \sum_{(v,u) \in \delta^-(u)} y_{vu} = 1, \quad \forall u \in V \tag{3.4}$$

$$y_{uv} \leq x_u, \quad \forall (u, v) \in A \tag{3.5}$$

$$x_u \in \{0, 1\}, \quad \forall u \in V \tag{3.6}$$

$$y_{uv} \in \{0, 1\}, \quad \forall (u, v) \in A \tag{3.7}$$

Ограничења (3.3) обезбеђују да ниједна два суседна чвора не могу истовремено бити део решења, чиме се гарантује да решење представља независан скуп. Ограничења (3.4) осигуравају да је сваки чвор u или укључен у решење, или га покрива тачно један његов сусед v који припада решењу. Даље, ограничења (3.5) проверавају да чвор u мора бити укључен у решење уколико покрива свог суседа v. У комбинацији са другим делом функције циља (3.2), ова ограничења омогућавају тачно рачунање тежина грана које повезују чворове ван решења са онима који су у решењу. Последња два скупа ограничења, (3.6) и (3.7), формално дефинишу претходно наведену тврдњу да су променљиве x и y бинарне.

Укупан број променљивих у овој формулацији износи $|V| + 2 \cdot |E|$, док укупан број ограничења износи $2 \cdot |V| + 5 \cdot |E|$.

3.3.2 ILP модел New-2

У другом ILP моделу, који је означен као New-2, примењена је Бендерова декомпозиција [13]. Ова техника је заснована на идеји подели-па-владај (енг. *divide-and-conquer*), где се скуп променљивих дели на два подскупа – тзв. мастер проблем се решава на првом подскупу, а потом се за дате вредности променљивих из првог подскупа решава потпроблем на другом подскупу. Ако се при решавању потпроблема покаже да одлуке донете у првој фази нису допустиве, додају се нова ограничења (Бендерова одсецања). У контексту разматраног проблема, променљиве y из модела New-1 се елиминишу, а уместо њих се уводе непрекидне променљиве q, које представљају тежину најјефтиније гране која повезује чвор ван решења са неким чвором у решењу.

Нека је $N'(u)$ низ суседних чворова чвора u, сортиран растуће по тежинама грана које их повезују са u. Уз тај додатак нотацији, оваква формулација, инспирисана претходним радовима на сличним проблемима [46, 6], омогућава да се MWIDS проблем изрази на следећи начин:

$$\min \sum_{u \in V} w_u x_u + q_u \tag{3.8}$$

уз услове

$$x_u + x_v \leq 1, \quad \forall e = \{u, v\} \in E \tag{3.9}$$

$$x_u + \sum_{v \in N(u)} x_v \geq 1, \quad \forall u \in V \tag{3.10}$$

$$q_u \geq w_{su} - \sum_{t=1}^{s-1} (w_{su} - w_{tu}) x_t - w_{su} x_u, \quad \forall u \in V, \forall s \in \{1, \ldots, |N'(u)|\} \tag{3.11}$$

$$x_u \in \{0, 1\}, \quad \forall u \in V \tag{3.12}$$

$$q_u \geq 0, \quad \forall u \in V \tag{3.13}$$

Ограничења (3.9), као и у моделу NEW-1, гарантују да решење чини независан скуп. Ограничења (3.10) обезбеђују да сваки чвор буде или укључен у решење, или доминиран од стране неког свог суседа који је у решењу. Трећи скуп ограничења (3.11) одговоран је за коректно израчунавање промењиве q_u и може се боље разумети разматрањем два случаја:

- Уколико је $x_u = 0$, тј. чвор u није део решења, ограничења постају слична Бендеровим оптималним одсецајућим равнима за проблеме попут простог локацијског проблема (енг. *Uncapacitated Facility Location Problem*, UFLP) [46] и проблема MWTDS, разматраног у поглављу 2. У том случају, променљива q_u преузима вредност тежине гране најмање тежине која повезује чвор u са неким чвором у решењу.

- Са друге стране, уколико је $x_u = 1$, тј. чвор u је део решења, десна страна неједнакости постаје највише нула, а самим тим и $q_u = 0$. Ово је у складу са дефиницијом проблема – цена постоји само за чворове који нису у решењу.

Ограничења (3.12) и (3.13) формално дефинишу тип променљивих: x су бинарне, док су q непрекидне и ненегативне.

Укупан број променљивих у овом моделу износи $2 \cdot |V|$, док је број ограничења једнак $3 \cdot |V| + 3 \cdot |E|$. Треба нагласити да број ограничења (3.11) одговара збиру степени свих чворова у графу, односно $2 \cdot |E|$, што произлази из чињенице да се за сваки чвор u генерише по једно ограничење за сваког његовог суседа $s \in N'(u)$. Број осталих ограничења се може једноставно израчунати.

3.4 Похлепни алгоритам за проблем MWIDS

Предложени похлепни алгоритам за решавање проблема MWIDS, означен као GREEDY-NEW, започиње од празног решења S. Затим се, у свакој итерацији, у решење додаје чвор v^* са највећом вредношћу функције оцене (енг. *score function*), која се дефинише као:

$$score(v) = \frac{\sum_{u \in N(v)} (w_u + \sum_{t \in N(u) \setminus \{v\}} w_{ut})}{w_v + \sum_{u \in N(v)} w_{uv}} \tag{3.14}$$

Притом, избор се врши из скупа свих чворова који до тог тренутка нису покривени, означеног са V'. Чвор се сматра покривеним уколико је или део решења S, или има бар једног суседа који је у S.

Пошто је циљ максимизовати вредност функције оцене, пожељно је да бројилац буде што већи, а именилац што мањи. Именилац представља суму тежина чвора v и свих њему суседних грана. Будући да се функција циља (3.1) минимизује, пожељно је да чвор v има малу тежину, а његови суседи велике. Стога се тежина чвора v налази у имениоцу, а тежине суседа у бројиоцу. Поред тога, у вредност функције циља улазе и тежине грана које повезују чворове ван решења са чворовима у решењу. Дакле, пожељно је да чвор v који се

додаје у решење има суседне гране мале тежине, те се и њихова сума налази у имениоцу. Са друге стране, повољно је да сви његови суседни чворови имају велике тежине суседних грана, осим гране која их повезује са самим чвором v. Овај део симболизује добитак који се постиже покривањем тих чворова помоћу v – пошто се они више не морају експлицитно покривати, њихова велика тежина и повезаност се неће појавити у вредности функције циља.

Важно је напоменути да се након додавања чвора v^* у решење, он и сви његови суседи уклањају из графа, јер су сада покривени. Конкретно, уклањају се сви чворови $u \in N[v^*]$ и све гране које су им суседне. Услед тога, ови чворови више ни на који начин не утичу на вредност функције оцене у наредним итерацијама, чиме се избегава вишеструко покривање истих чворова.

Псеудокод је приказан у алгоритму 6. У свакој итерацији се избацује бар један чвор из графа, па је број итерација у најгорем случају једнак $|V|$. У оквиру једне итерације, за сваки чвора од преосталих чворова $v \in V'$, потребно је проћи кроз све његове суседе $N_{G'}(v)$, што укупно даје $\sum_{v \in V'} deg_{G'}(v) = O(|E|)$, будући да је, како је већ поменуто, збир степени свих чворова у графу пропорционалан броју грана. Дакле, укупна сложеност алгоритма је $O(|V| \cdot |E|)$.

Алгоритам 6 GREEDY-NEW

Input: граф $G = (V, E)$
Output: решење S

$S \leftarrow \emptyset$
$G' = (V', E') \leftarrow G = (V, E)$
$edgeWeights \leftarrow [\sum_{v \in N(u)} w_{uv} | u \in V]$
while $size(V') > 0$ **do**
    $v^* \leftarrow -1$
    $scoreMax \leftarrow -1$
    for $v \in V'$ **do**
        $nodeWeights \leftarrow 0$
        $uvEdgeWeights \leftarrow 0$
        $otherEdgeWeights \leftarrow 0$
        for $u \in N_{G'}(v)$ **do**
            $nodeWeights \leftarrow nodeWeights + w_u$
            $uvEdgeWeights \leftarrow uvEdgeWeights + w_{uv}$
            $otherEdgeWeights \leftarrow otherEdgeWeights + edgeWeights_u - w_{uv}$
        end for
        $score \leftarrow (nodeWeights + otherEdgeWeights)/(w_v + uvEdgeWeights)$
        if $score > scoreMax$ **then**
            $scoreMax \leftarrow score$
            $v^* \leftarrow v$
        end if
    end for
    $S.add(v^*)$
    for $v \in N_{G'}[v^*]$ **do**
        for $u \in N_{G'}(v)$ **do**
            $edgeWeights_u \leftarrow edgeWeights_u - w_{uv}$
        end for
        remove node v from G'
    end for
end while
return S

3.5 Експериментални резултати

Квалитет предложених ILP модела и похлепног алгоритма је процењен на скупу инстанци који је претходно установљен у литератури. Резултати су упоређени са осам постојећих приступа описаних у одељку 3.2. Све методе су имплементиране у програмском језику Python. Експерименти су извршени на рачунару са Intel Core i9-11900 @ 2.5GHz процесором, уз ограничење од 4GB RAM меморије по извршавању, под оперативним системом Ubuntu 22.04. ILP модели су решавани коришћењем `CPLEX` решавача верзије 20.1.

3.5.1 Инстанце проблема

Коришћен је исти скуп референтних инстанци као у досадашњим радовима [37, 110]. У овом скупу постоје две врсте графова:

- Случајни графови – генерисани коришћењем Ердош-Ренји модела [42], где је

вероватноћа гране између свака два чвора означена са *ep*. Дакле, параметар *ep* одређује густину графа. Коришћене су вредности $ep \in \{0.05, 0.15, 0.25\}$, и тако су добијени графови различитих густина.

- Случајни геометријски графови су добијени тако што се сваком чвору додељују случајне координате унутар јединичног квадрата, а чворови се повезују ако је растојање између њих мање од задатог радијуса r. Ова метода је позната као модел јединичног диска (енг. *Unit disk model*) [29] и често се користи за симулацију ад-хок бежичних мрежа [44]. У овом случају, густина графа зависи од вредности радијуса r, а коришћене вредности су $r \in \{0.14, 0.24, 0.34\}$, одабране са циљем приближног поклапања густине са случајним графовима.

За обе врсте, генерисани су графови различитих величина – $|V| \in \{100, 500, 1000\}$.

Тежине чворова и грана су генерисане на три начина:

1. И чворови и гране имају тежине изабране из униформне расподеле $\mathcal{U}\{0, 100\}$. Ови графови се називају неутрални (енг. *neutral graphs*, NG).

2. Тежине чворова су изабране из расподеле $\mathcal{U}\{0, 1000\}$, а тежине грана из расподеле $\mathcal{U}\{0, 10\}$. На овај начин су добијени графови у којима је одабир чворова важнији од одабира грана, те се називају графови оријентисани на чворове (енг. *vertex oriented graphs*, VG).

3. И обрнуто, тежине чворова су изабране из расподеле $\mathcal{U}\{0, 10\}$, а тежине грана из расподеле $\mathcal{U}\{0, 1000\}$. У оваквим графовима је одабир грана кључан, па се називају графови оријентисани на гране (енг. *edge oriented graphs*, EG).

За сваку комбинацију типа графа, броја чворова, густине (параметар *ep* или r), и шеме тежина, генерисано је по 10 инстанци. Укупан број инстанци износи 540, од чега је 270 случајних и 270 случајних геометријских графова.

Инстанце и имплементација предложених метода јавно су доступне на адреси `https://github.com/StefanKapunac/mwids_public`.

3.5.2 Резултати

Резултати су представљени у четири табеле: за сваки тип графова (случајни или геометријски) и за сваку класу алгоритама (егзактни или хеуристички). Табеле 3.1 и 3.2 приказују резултате егзактних метода за случајне и случајне геометријске графове, тим редом. Табеле 3.3 и 3.4 садрже резултате хеуристичких метода. Структура свих табела је иста. Прве три колоне у табелама означавају број чворова $|V|$, шему тежина и параметар густине (*ep* или r). Наредне колоне представљају просечне вредности добијене за 10 инстанци по групи: *result* – просечан број чворова у решењу, *time*[2] – просечно време извршавања у секундама, а за ILP моделе додатно и *gap* – просечна разлика између добијеног решења и најбоље могуће вредности у процентима (енг. *optimality gap*). У сваком реду су најбољи резултати подебљани, а најбољи међу похлепним хеуристикама су подвучени.

На основу експерименталних резултата егзактних метода, могу се извући следећи закључци:

- Модели New-1 и New-2 постижу боље резултате у великој већини случајних графова у поређењу са остала три ILP модела. Поред тога, ови модели оптимално

[2]Информације о просечном времену извршавања за ILP моделе из рада [37] нису биле доступне.

Табела 3.1: Нумерички резултати егзактних метода на случајним графовима

$\|V\|$	weight scheme	ep	ILP-1 result	ILP-1 gap	ILP-2 result	ILP-2 gap	ILP-3 result	ILP-3 gap	NEW-1 result	NEW-1 gap	NEW-1 time	NEW-2 result	NEW-2 gap	NEW-2 time
100	NG	0.05	**3049.8**	0.7	**3049.8**	0	3051.9	0.5	**3049.8**	0	0.2	**3049.8**	0	0.1
		0.15	2445.2	30.4	2396.3	28.3	2398.4	44.6	**2330.2**	0	2.2	**2330.2**	0	2.5
		0.25	2161.1	31.3	2114.3	24.3	2167.6	58.8	**2069.3**	0	8.3	**2069.3**	0	8.8
	VG	0.05	**7715.4**	0	**7715.4**	0	**7715.4**	0	**7715.4**	0	0.2	**7715.4**	0	0.1
		0.15	**3046.6**	0	**3046.6**	0	**3046.6**	0	**3046.6**	0	0.8	**3046.6**	0	0.5
		0.25	**1808.4**	0	**1808.4**	0	**1808.4**	0	**1808.4**	0	1.1	**1808.4**	0	0.5
	EG	0.05	**14378.7**	0	**14378.7**	0	**14378.7**	0	**14378.7**	0	0.4	**14378.7**	0	0.1
		0.15	15473	42.4	15198.9	39.6	15098.3	57.7	**14563.3**	0	17.7	**14563.3**	0	11.1
		0.25	16001.7	61.5	14557.7	28.5	15346.3	69.3	**14382.2**	0	35.3	**14382.2**	0	28.5
500	NG	0.05	11857.9	51.9	11809.3	50.5	12882.6	91.7	**10869.1**	30.7	TLE	11163.6	32.5	TLE
		0.15	10050.1	69.4	9928.8	68.1	13196.7	98.1	9568.7	55.1	TLE	**9348.5**	54.4	TLE
		0.25	11341.1	78	9465.7	73.9	12722.4	99	**9078.9**	62.5	TLE	9099.1	62.5	TLE
	VG	0.05	12557.6	52.6	11403	47.1	12059.3	58	10529.8	34.7	TLE	**10048.3**	29.5	TLE
		0.15	5940.1	61.3	6122.4	59.3	5474.6	80.5	4815.5	48.7	TLE	**4107.2**	31.3	TLE
		0.25	8166.2	79.4	10145.8	82.9	3628.7	85.6	3514.6	45.1	TLE	**2749.4**	14.3	TLE
	EG	0.05	97915.3	82.9	89580.7	81.1	107463.5	98.7	87091.4	70.5	TLE	**83820.5**	69.6	TLE
		0.15	107834.7	93.4	91564.1	91.7	117036.1	99.9	**84871.4**	84	TLE	86965	84.5	TLE
		0.25	100750.3	95	88687.7	94	113798	99.9	**87735.9**	87.6	TLE	100113	89	TLE
1000	NG	0.05	25156.1	69.5	25986	68.7	27158.7	97	**21151.2**	51.9	TLE	21308.9	52.5	TLE
		0.15	21117.8	81.1	**18282.9**	76.5	22984.5	99.1	18554.6	68.1	TLE	21105.9	71.9	TLE
		0.25	158097.1	93.2	88053.8	88.7	21821.5	99.5	**17600.2**	73.1	TLE	19726.8	75.9	TLE
	VG	0.05	35766.3	83.3	39356.1	83.3	15464.1	77.4	14648.2	56.7	TLE	**13056.9**	50.4	TLE
		0.15	35678.2	91.3	35904.3	97	11586.3	92.3	6596.4	51.8	TLE	**5733.7**	42.4	TLE
		0.25	35838.9	96.5	22569.5	100	11674.7	96.7	7311	60.7	TLE	**4242.6**	38.2	TLE
	EG	0.05	209391.8	91.1	198540.7	90.1	229778.7	99.7	**190225.4**	84.7	TLE	197814.4	85.7	TLE
		0.15	832437.1	97.7	**191911.2**	96.2	229888.1	100	304074.6	94.7	TLE	194334.6	91.8	TLE
		0.25	1128240.1	98.6	2041102.4	99.7	221492	100	363299.1	97.1	TLE	**186606.1**	93.2	TLE

Табела 3.2: Нумерички резултати егзактних метода на случајним геометријским графовима

$\|V\|$	weight scheme	ep	ILP-1 result	ILP-1 gap	ILP-2 result	ILP-2 gap	ILP-3 result	ILP-3 gap	NEW-1 result	NEW-1 gap	NEW-1 time	NEW-2 result	NEW-2 gap	NEW-2 time
100	NG	0.14	**3261.1**	0	**3261.1**	0	**3261.1**	0	**3261.1**	0	<0.1	**3261.1**	0	<0.1
		0.24	2942.9	21.7	2917.5	15.6	2884.5	3	**2882.5**	0	0.6	**2882.5**	0	0.4
		0.34	2878.5	26.5	2841.8	8.6	2876.7	17.6	**2828**	0	1.3	**2828**	0	1.5
	VG	0.14	**5731.8**	0	**5731.8**	0	**5731.8**	0	**5731.8**	0	<0.1	**5731.8**	0	<0.1
		0.24	**1981.8**	0	**1981.8**	0	**1981.8**	0	**1981.8**	0	<0.1	**1981.8**	0	<0.1
		0.34	**940.5**	0.3	**940.5**	0	**940.5**	0	**940.5**	0	0.2	**940.5**	0	<0.1
	EG	0.14	**19179.4**	0	**19179.4**	0	**19179.4**	0	**19179.4**	0	<0.1	**19179.4**	0	<0.1
		0.24	22519.7	20.5	22325.5	16.5	**22065.9**	8.2	**22065.9**	0	0.6	**22065.9**	0	0.4
		0.34	24295.9	31.1	**23717.6**	7	24009.2	19	**23717.6**	0	1.4	**23717.6**	0	1.7
500	NG	0.14	14942.7	56	15016.4	54.9	15457.2	85.1	13726.8	15.7	TLE	**13565**	12.7	TLE
		0.24	19028.3	72.7	15668.7	66.5	16788.4	95.5	13375.1	27.8	TLE	**13352.3**	28.3	TLE
		0.34	18602.1	75.8	15468.3	67.8	18948.1	97.6	13379.3	29.6	TLE	**13253.1**	29.5	TLE
	VG	0.14	**4377**	0.4	**4377**	0	4381.6	1.4	**4377**	0	4.9	**4377**	0	0.6
		0.24	2619.3	7.6	2596.4	5.8	2665.1	39	**2573.7**	0	49.3	**2573.7**	0	5.9
		0.34	3933.7	51.2	2205.3	17	2305.3	63.8	**2181.6**	0	294	**2181.6**	0	12.4
	EG	0.14	128902.8	66.2	128784.8	65.7	129521.2	91.8	116047.1	21.1	TLE	**115232**	19.7	TLE
		0.24	175979.6	76.6	147009	71.5	162967.5	98.3	124047.4	31.1	TLE	**122879.2**	31.5	TLE
		0.34	180632.5	78.4	149305.9	71.2	188131.4	99	**125499.8**	31.2	TLE	126997.4	33	TLE
1000	NG	0.14	38289.8	73.1	33123.7	68.5	38904.3	95.8	27649.2	30.7	TLE	**27066.6**	29.6	TLE
		0.24	61533.9	86.9	**33744.7**	79.7	38610.1	98.3	33758.1	48	TLE	35032.5	50	TLE
		0.34	121566.1	100	48329.9	95.9	38549.2	98.9	50289.4	100	TLE	**34725.5**	49.6	TLE
	VG	0.14	6071.9	11.6	6614.5	14.1	6144.7	43.9	**5830.7**	0	335.9	**5830.7**	0	21.7
		0.24	11493	64.6	12485.5	92.3	4528.1	74.7	4356.6	10	TLE	**4279.4**	0	669.7
		0.34	17662.5	100	9742.4	100	6214.7	88.7	4211.3	20.8	TLE	**3975.1**	2	2085.2
	EG	0.14	362985.2	78.8	311510.5	75.5	350946.2	98.4	**252929.5**	35.5	TLE	255236.5	36.6	TLE
		0.24	347041.7	88.6	**326670.5**	83.2	375956.1	99.4	375018.2	62.4	TLE	362021.8	56.7	TLE
		0.34	1232171.4	99.7	476998.4	97.3	370791.1	99.6	498328.3	100	TLE	**360711.3**	56.2	TLE

решавају све инстанце са 100 чворова, за разлику од модела из [37], који постижу оптималност у само 4 или 5 од укупно 9 група инстанци са тим бројем чворова. Просечно време извршавања модела New-1 и New-2 за инстанце са 100 чворова је кратко, где се приближно половина решава за мање од једне секунде, а максимално време достиже за групу најгушћих графова оријентисаних на гране, где траје 35.3 секунде.

- Модели New-1 и New-2 такође надмашују резултате сва три конкурентска ILP модела на већини случајних геометријских графова. Оба модела успешно решавају све инстанце са 100 чворова. Поред тога, достижу оптималност и на свим графовима оријентисаним на чворове величине 500, као и на неким величине 1000. Међу тим највећим графовима са 1000 чворова који су оријентисани на чворове, New-1 решава групу најмање густине ($r = 0.14$), док New-2 решава скоро све инстанце оријентисане на чворове, осим три из најгушће групе ($r = 0.34$). Важно је напоменути и да су времена извршавања модела New-1 и New-2 у неким случајевима значајно мања од метахеуристичких метода, поготово на инстанцама мање густине.

Табела 3.3: Нумерички резултати хеуристичких метода на случајним графовима

$\|V\|$	weight scheme	ep	Greedy-1		Greedy-2		Greedy-New		Pbig		Cmsa-Pbig		Lsrr	
			result	time	result	time	result	time	result	time	result	time	result	time
100	NG	0.05	3589.1	<0.1	3519.1	<0.1	3340.2	<0.1	**3049.8**	0.54	**3049.8**	8	**3049.8**	<0.1
		0.15	3014.4	<0.1	2981.3	<0.1	2768.3	<0.1	2330.9	28.31	2338.1	48.3	**2230.2**[*]	0.15
		0.25	2883.5	<0.1	2796.1	<0.1	2577.4	<0.1	2070.9	0.16	2093.9	54.1	**2069.3**	0.66
	VG	0.05	10465.6	<0.1	11756.6	<0.1	10785.4	<0.1	7747	76.47	7860	59.4	**7715.4**	0.13
		0.15	4891.6	<0.1	5845.4	<0.1	4941.6	<0.1	3050.3	16.9	3070.3	40.6	**3046.6**	0.41
		0.25	3297.5	<0.1	3488.9	<0.1	2962.6	<0.1	**1808.4**	3.5	**1808.4**	15.5	**1808.4**	0.79
	EG	0.05	25698.7	<0.1	22269.3	<0.1	19976.7	<0.1	**14378.7**	0.78	**14378.7**	37.9	**14378.7**	<0.1
		0.15	27528.4	<0.1	23404.5	<0.1	20973	<0.1	14687.8	0.19	**14563.3**	17.2	**14563.3**	<0.1
		0.25	25451.4	<0.1	21770	<0.1	21302.8	<0.1	14506.6	<0.1	**14382.2**	37.7	**14382.2**	<0.1
500	NG	0.05	14143.1	<0.1	13535.1	<0.1	12272	0.4	10327.3	127.37	10140.6	667.3	**10041.5**	619.14
		0.15	12268.5	<0.1	11558	<0.1	10793.8	0.4	8297.5	47.33	8046.2	688.2	**7989**	869.88
		0.25	11630.3	<0.1	10429.5	0.1	10038.2	0.5	7633.7	10.16	7443	538.6	**7374.3**	841.53
	VG	0.05	15501.5	<0.1	18298.1	<0.1	14745.2	0.3	9822.6	924.21	9588.2	912.1	**8421.1**	848.96
		0.15	6496.3	<0.1	7300.1	<0.1	6021.8	0.4	3581.7	219.52	3557.8	599.8	**3497**	815.76
		0.25	4212.4	<0.1	4463.7	0.1	4112.9	0.6	2590.6	52.82	2586.5	521.1	**2572.5**	725.55
	EG	0.05	125357.6	<0.1	108178	<0.1	100325.8	0.4	70028.6	1008.89	67528.9	713.6	**65441.3**	599.88
		0.15	114951	<0.1	102365.1	<0.1	90468.8	0.6	64673.8	109.71	62950.1	798.1	**61456.5**	930.79
		0.25	111012.3	<0.1	99018.2	0.1	90066.8	0.5	64112.7	33.26	61411.1	294.9	**60637.4**	769.72
1000	NG	0.05	25569.6	<0.1	23489.7	0.1	22880	2.4	17723.7	1750.12	17819.4	1262.1	**17478**	1985.83
		0.15	20827.1	<0.1	20689.1	0.2	19301.5	2	14731.3	260.52	14461.9	813.3	**14340.7**	1762.62
		0.25	20858.8	<0.1	19280.5	0.4	18828.8	2.4	13968.5	340.14	13685.6	1480.6	**13397.9**	1688.65
	VG	0.05	18048.6	<0.1	20142.3	0.1	17464	1.4	11301.7	2018.01	11034.6	1363.5	**10392.2**	1690.35
		0.15	7408.3	<0.1	7987.4	0.2	6790.6	2.7	4540.3	695.71	4456.4	1362.7	**4440.7**	1638.47
		0.25	4941.9	<0.1	5566.6	0.4	4802.3	2.4	3519	306.67	3460.4	1049.3	**3454**	1610.84
	EG	0.05	238600	<0.1	202992	0.1	192329.5	1.7	133667.8	2121.82	130889.2	1786.2	**127796**	1906.56
		0.15	209709.3	<0.1	182726.6	0.2	176920.4	2	123760.9	99.67	120997.2	1478.6	**119250.3**	1781.76
		0.25	198537	<0.1	181150	0.4	170618.2	2.4	124193.3	447.95	120288.7	1359.1	**114124.2**	1793.35

[*] Делује да је дошло до штампарске грешке у раду [110], пошто је просек оптималних решења пронађених новим ILP моделима већи, али се састоји од сличних цифара (2330.2).

Табела 3.4: Нумерички резултати хеуристичких метода на случајним геометријским графовима

$\|V\|$	weight	ep	Greedy-1		Greedy-2		Greedy-New		Pbig		Cmsa-Pbig		Lsrr	
	scheme		result	time	result	time	result	time	result	time	result	time	result	time
100	NG	0.14	3870.6	<0.1	3646.4	<0.1	3506.2	<0.1	**3261.1**	11.9	**3261.1**	3.3	**3261.1**	0.1
		0.24	3798.8	<0.1	3378.1	<0.1	3248.7	<0.1	**2882.5**	3	**2882.5**	2.7	**2882.5**	<0.1
		0.34	3766.6	<0.1	3388.1	<0.1	3175.6	<0.1	**2828**	0.7	**2828**	27.9	**2828**	<0.1
	VG	0.14	7364.9	<0.1	7514.3	<0.1	7471.2	<0.1	**5731.8**	12.4	5740	2	**5731.8**	<0.1
		0.24	2880.1	<0.1	2724.2	<0.1	2808.3	<0.1	**1981.8**	<0.1	**1981.8**	2.2	**1981.8**	<0.1
		0.34	1741	<0.1	1832.3	<0.1	1718.6	<0.1	968.8	<0.1	**940.5**	1.2	**940.5**	<0.1
	EG	0.14	29011.6	<0.1	24998.3	<0.1	21463.6	<0.1	19313.2	117.6	**19179.4**	10.1	**19179.4**	<0.1
		0.24	35312.1	<0.1	28647.1	<0.1	26572.3	<0.1	22108.3	6.4	**22065.9**	5.5	**22065.9**	<0.1
		0.34	37929.4	<0.1	30503.4	<0.1	27020.8	<0.1	23900	1.1	**23717.6**	76.7	**23717.6**	<0.1
500	NG	0.14	18408.3	<0.1	16208.4	<0.1	15574.6	0.2	13341.5	835.4	13301.2	548.6	**13269.4**	551.1
		0.24	18548.8	<0.1	15882.9	<0.1	15613.7	0.2	12943.1	195.7	12783.3	129.6	**12764**	180.1
		0.34	18311.4	<0.1	15497.8	0.1	16497.8	0.3	13065.5	6.2	12954.8	327.5	**12887.5**	86.1
	VG	0.14	6807.8	<0.1	7087.8	<0.1	6974.8	0.2	4400.9	513.7	4389.7	165.8	**4377**	554.1
		0.24	3526.6	<0.1	3121.1	<0.1	3553.5	0.2	**2573.7**	9.7	**2573.7**	17.8	**2573.7**	3.6
		0.34	2632.2	<0.1	2830.7	0.1	2438	0.3	**2181.6**	7.2	**2181.6**	13.1	**2181.6**	2
	EG	0.14	177816.7	<0.1	148267	<0.1	136384.9	0.2	112404.8	731	111638.2	680.5	**110988.7**	190.3
		0.24	181739.2	<0.1	149980.2	<0.1	148916.9	0.2	118765.6	16.3	117439.8	405.9	**117020.2**	23.5
		0.34	190996.4	<0.1	155128.6	0.1	159464	0.3	122977.3	5.2	120883.7	346.4	**120381.3**	31.4
1000	NG	0.14	36214.6	<0.1	32393.4	0.1	31882	0.6	25892.9	776.7	25719	1695.2	**25563.3**	1792.2
		0.24	36750.4	<0.1	31462.5	0.2	31698.7	0.9	25570.6	524.8	25138.9	1081.1	**25111.7**	1388.1
		0.34	36913.2	<0.1	30752.1	0.3	33603.5	1.2	25714.2	17.2	25345	1698	**25114.3**	704.2
	VG	0.14	8552.3	<0.1	8613.7	0.1	8193.4	0.7	5869	1778.5	5890.4	950.4	**5830.7**	1244.3
		0.24	4977.4	<0.1	5146.2	0.2	4901.6	0.9	4281.2	109.1	4283.6	90.1	**4279.4**	177.8
		0.34	4656.5	<0.1	4693.1	0.4	4417.7	1.2	3974.5	63.4	3974.3	264.2	**3970.8**	44.2
	EG	0.14	358550.4	<0.1	305034.7	0.1	289269.1	0.7	236226.6	2582.6	233127.8	1711.2	**230030.4**	1327
		0.24	357735.8	<0.1	295099.4	0.2	307039.9	0.9	239560.7	451.8	238364.5	1277.3	**236439.8**	907
		0.34	369051.9	<0.1	303712.9	0.3	323414.4	1.2	244685.7	24.3	246171.8	1195.2	**243178.8**	793.7

Резултати хеуристичких метода указују на следеће:

- Алгоритам Greedy-New у већини случајева надмашује резултате постојећих похлепних алгоритама на скоро свим случајним графовима. Једина два изузетка су групе инстанци оријентисаних на чворове величине 100 са густином $ep = 0.25$ и 500 са густином $ep = 0.05$ и $ep = 0.15$, где Greedy-1 постиже нешто боље резултате.
- На случајним геометријским графовима ситуација није толико убедљива. Ипак, у просеку, Greedy-New постиже боље резултате од осталих похлепних метода.
- Метахеуристике постижу боље резултате од похлепних метода, нарочито на већим графовима, при чему жртвују време извршавања у корист квалитета решења. Посебно се истиче алгоритам Lsrr као тренутно најбоље решење, који постиже оптималне резултате у свим случајевима где је оптимум познат, достижући резултате предложених ILP модела.

3.5.3 Статистичка анализа

Примењена је иста статистичка методологија као и у поглављу 2, која се састоји од две фазе. У првој фази су сви алгоритми анализирани заједнички применом Фридмановог теста [47]. Уколико би нулта хипотеза, која претпоставља статистичку једнакост поређених метода, била одбачена, у другој фази би се применио Неменјијев пост-хок тест [93].

Резултати добијени овом анализом приказани су на слици 3.3 у облику графика критичне разлике (енг. *critical difference*, CD). Као што је раније поменуто, на овим графицима сваки алгоритам је позициониран дуж хоризонталне осе на основу просечног ранга који је постигао на анализираном скупу инстанци. Критична разлика, рачуната за ниво значајности 0.05, између два алгоритма представља минималну разлику у просечном

рангу која је потребна да би се сматрали статистички различитим. Статистичка једнакост два алгоритма је на графику представљена црном хоризонталном линијом између њих.

Анализом графика са слике 3.3 може се закључити следеће:

- Метахеуристички приступи имају најнижи просечни ранг, што указује на квалитетнија решења у односу на преостале методе. Посебно се истиче LSRR алгоритам, који је остварио најнижи просечни ранг међу свим поређеним методама. Ипак, када се све инстанце узму у обзир, не постоји статистички значајна разлика између резултата добијених помоћу три метахеуристике (PBIG, CMSA-PBIG и LSRR) и ILP модела NEW-2. Што се тиче модела NEW-1, само LSRR показује статистички значајно боље резултате. Међу похлепним алгоритмима, GREEDY-NEW има најнижи ранг и статистички значајно је бољи од GREEDY-2, али не и од GREEDY-1.

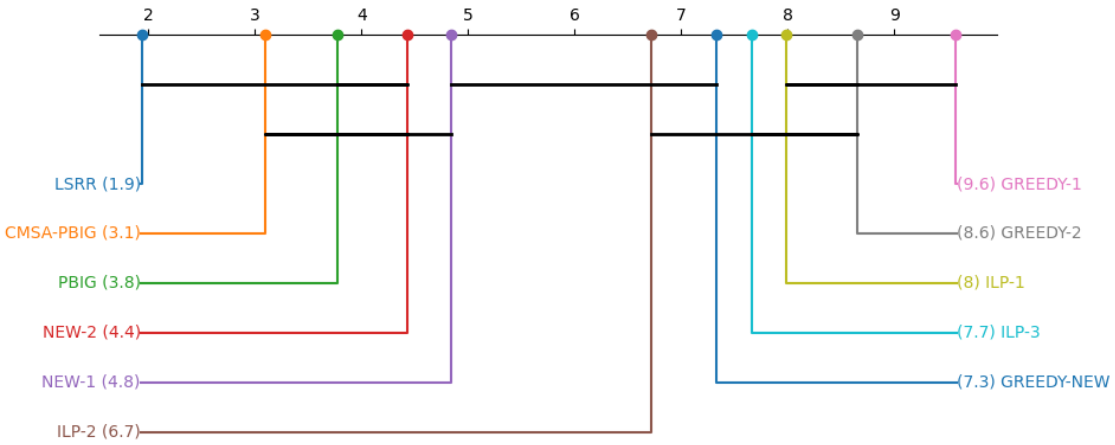
- Када се посматрају само егзактне методе, модели NEW-1 и NEW-2 се издвајају и статистички значајно надмашују преостала три ILP модела. Притом, међу новопредложеним моделима нема статистички значајне разлике. Такође, ни међу моделима ILP-1, ILP-2 и ILP-3 не постоји статистички значајна разлика.

- При анализи само похлепних метода, GREEDY-NEW је статистички значајно бољи од GREEDY-1 и GREEDY-2. Иако GREEDY-2 има нешто нижи просечни ранг од GREEDY-1, не постоји статистички значајна разлика између њих.

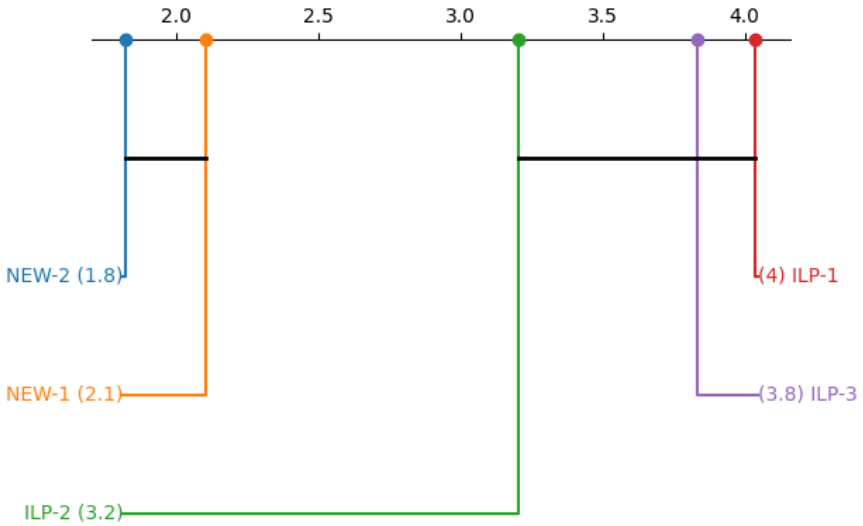
Поред претходне анализе на свим инстанцама, спроведена је и појединачна анализа на различитим типовима графова из коришћеног скупа инстанци. Графици критичне разлике посебно за случајне и случајне геометријске графове приказани су на слици 3.4. На основу ових графика, могу се извући следећи закључци:

- При анализи групе случајних графова, алгоритам LSRR се истиче као једина метода која је статистички значајно боља од оба предложена ILP модела – NEW-1 и NEW-2. Као и раније, не постоји статистички значајна разлика међу три метахеуристике. Иако GREEDY-NEW има нижи просечни ранг од ILP-1, ILP-2, ILP-3, GREEDY-1 и GREEDY-2, та разлика није статистички значајна.

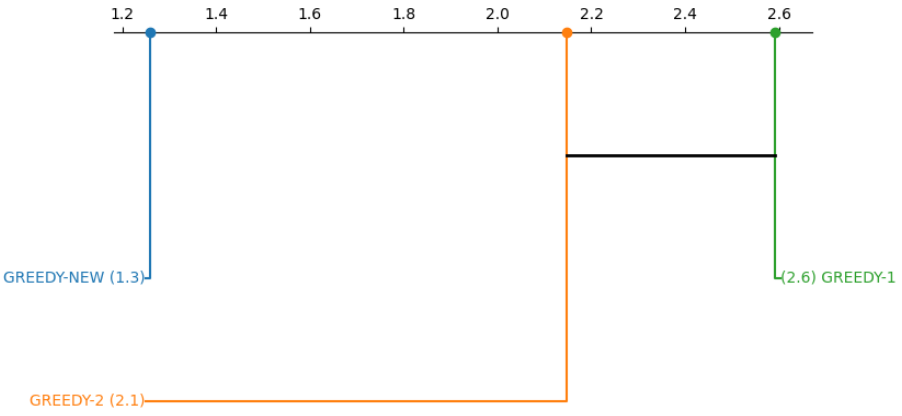
- У случају случајних геометријских графова, не постоји статистички значајна разлика између предложених ILP модела (NEW-1 и NEW-2) и метахеуристичких метода (LSRR, PBIG и CMSA-PBIG), иако и у овом случају метахеуристике имају нижи просечни ранг. Похлепни алгоритам GREEDY-NEW поново постиже боље просечне резултате од осталих похлепних метода, али та разлика није статистички значајна.



(а) Сви приступи

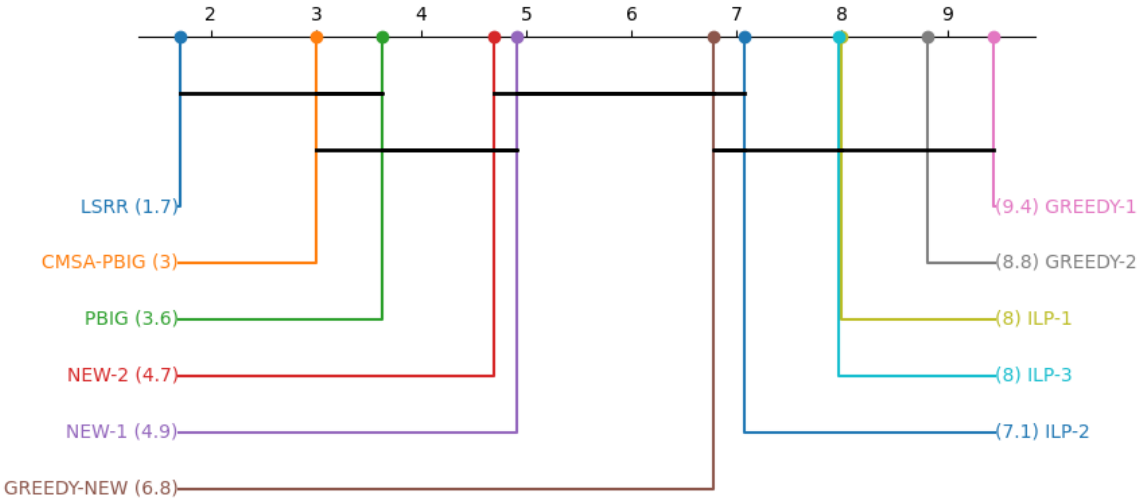


(б) Егзактне методе

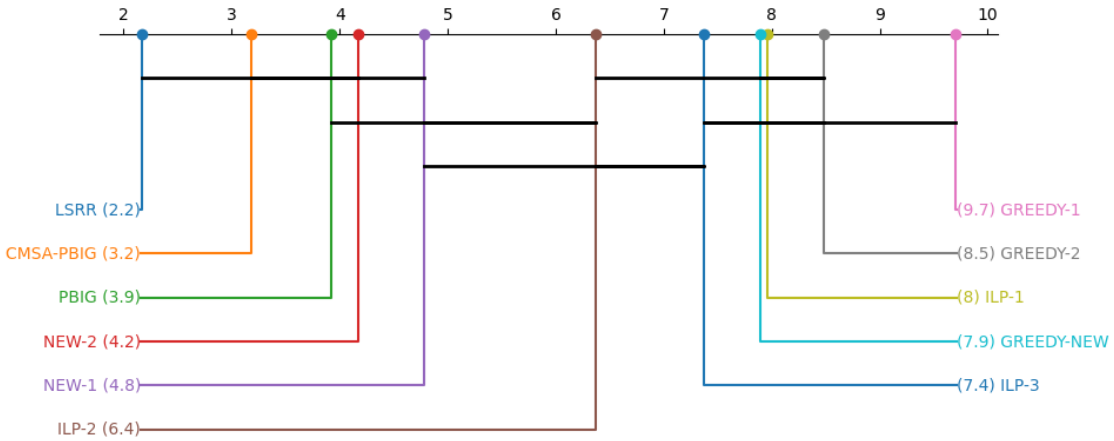


(в) Похлепне хеуристике

Слика 3.3: Графици критичне разлике за све инстанце из два референтна скупа



(а) Случајни графови



(б) Случајни геометријски графови

Слика 3.4: Графици критичне разлике за две групе инстанци

3.6 Завршна разматрања

У оквиру овог поглавља предложена су два ILP модела и похлепна хеуристика за решавање проблема минималне тежинске независне доминације. ILP модели показују се као веома ефикасни на малим и средњим инстанцама, где углавном надмашују постојеће егзактне приступе. Поред тога, постижу конкурентне перформансе у односу на метхеуристичке приступе из литературе и на одређеним класама великих инстанци. Развијени похлепни алгоритам показује супериорне перформансе у односу на конкурентске похлепне приступе, а на већим инстанцама надмашује и неке ILP моделе. Предложени приступи представљају најбоље доступне методе у својим категоријама, егзактних, односно похлепних алгоритама, што их чини погодним градивним блоковима за будуће метахеуристичке и хибридне методе.

Поглавље 4

Проблем k-јаке римске доминације

Још једна значајна класа доминацијских проблема су проблеми римске доминације. Основни проблем римске доминације има занимљиву историјску позадину инспирисану војном стратегијом Римског царства. Наиме, Константин Велики је осмислио ефикасан начин одбране великог царства пажљивим распоређивањем легија по различитим регионима. Регион се сматрао безбедним уколико се на његовој територији налазила бар једна легија, док су области без иједне легије биле небезбедне. Таква небезбедна подручја су штићена премештањем једне легије из суседног обезбеђеног региона. Међутим, премештање легије није дозвољено ако би полазни регион остао необезбеђен. Другим речима, легија се могла преместити у суседни регион само ако су у почетном региону постојале бар две легије.

Овај проблем је формално уведен у раду [32], а уследиле су разне варијације и, како практична, тако и теоријска истраживања [26, 27, 12, 1, 4]. Један од проблема из ове класе је проблем k-јаке римске доминације (енг. *k-Strong Roman Domination Problem*, k-SRDP), који је предложен у раду [84]. Овде се разматра k истовремених напада, где је k број већи од један. Било који подскуп величине k може бити нападнут и у сваком од тих случајева потребно је да постоји план одбране. Такав сценарио је реалистичан и проналази примену у ситуацијама као што су: борба против тероризма, управљање у ванредним ситуацијама и поремећајима у ланцу снабдевања. Тежина овог проблема произлази из чињенице да је за саму проверу допустивости решења потребно експоненцијално много корака у општем случају.

Садржај овог поглавља је базиран на раду [39].

4.1 Дефиниција проблема

Као и до сада, проблем је дефинисан на неусмереним графовима, где је са $d(v)$ означен степен чвора, а са $\Delta(G) = \max\{d(v) \mid v \in V\}$ максималан степен у графу. Број јединица, тј. легија, у сваком чвору се моделује функцијом $f\colon V \mapsto \mathbb{N}_0$. Дакле, $f(v)$ представља број јединица у чвору v. Укупан број јединица у целом графу је $\omega(f, G) = \sum_{v \in V} f(v)$. Сваки чвор за који је $f(v) = 0$ је небрањен и њега мора да обрани један од суседа.

За фиксиран параметар k напад се дефинише као подскуп чворова кардиналности k, односно формално $P := \{v_1, v_2, \ldots, v_k\} \subseteq V$. Број различитих подскупова величине k је $\binom{|V|}{k}$, а скуп свих могућих напада се означава са $\mathcal{P}(G) := \{P_1, \ldots, P_{\binom{|V|}{k}}\}$. У зависности од параметра k број различитих напада може да расте и експоненцијално са величином графа, што чини овај проблем изазовним.

Циљ је пронаћи што ефикаснију, у смислу укупног броја јединица, стратегију распоређивања јединица по чворовима графа, која гарантује одбрану од сваког могућег напада. Формално, функција $f : V \mapsto \{0, 1, \ldots, \min(\Delta(G), k) + 1\}$ је валидна k-SRD функција ако је сваки могући напад из $\mathcal{P}(G)$ одбрањен, уз следећа правила одбране:

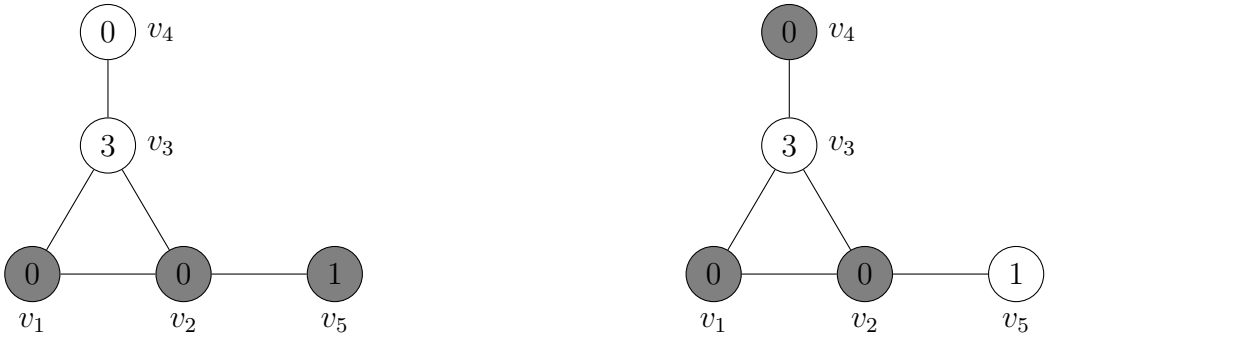
- Чвор са $f(v) = 1$ може да брани само себе.
- Чвор са $f(v) \geq 2$ може да брани себе и највише $f(v) - 1$ суседних чворова који су нападнути, а имају вредност 0.
- Сваки нападнути чвор са $f(v) = 0$ мора бити одбрањен од стране бар једног суседа.

Потребно је пронаћи такву функцију са минималним укупним бројем јединица $\omega(f, G)$. Број јединица у оптималном решењу се означава са $\gamma_{k\text{-}SRD}(G)$ и назива се k-SRD бројем графа, а функција за коју се та вредност достиже се назива $\gamma_{k\text{-}SRD}(G)$ функцијом.

Како је сваки чвор коме је додељена вредност већа од 0 одбрањен, тривијално решење за било које k јесте додела вредности 1 сваком чвору. У том случају, сваки чвор брани самог себе, па је испуњен услов да су сви чворови одбрањени без обзира на то који је напад у питању. Дакле, број чворова у графу $|V|$ представља горњу границу за $\gamma_{k\text{-}SRD}(G)$, па су интересантна само она решења у којима је бар један чвор означен са 0.

4.1.1 Пример инстанце

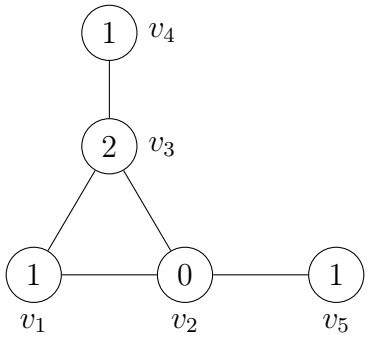
На слици 4.1 приказан је једноставан граф за напад величине 3. Нападнути чворови су обојени сивом бојом. Бројеви унутар чворова представљају број јединица у том чвору, односно вредности функције f. Укупан број распоређених јединица је 4. Међутим, функција f са ове слике није валидна k-SRD функција, јер постоје напади које није могуће одбранити.



(а) Одбрањив напад $P = \{v_1, v_2, v_5\}$ (б) Неодбрањив напад $P = \{v_1, v_2, v_4\}$

Слика 4.1: Пример инстанце и субоптималне стратегије за 3-SRD

Са друге стране, на слици 4.2 је приказано оптимално решење за ту инстанцу. Укупан број искоришћених јединица је 5, али је сада могуће одбранити сваки напад.



Слика 4.2: Оптимална стратегија за 3-SRD

4.2 Преглед литературе

Као што је већ поменуто, проблем је уведен у раду [84]. Аутори су предложили два егзактна алгоритма: ILP модел и приступ заснован на Бендеровој декомпозицији. Број променљивих у ILP моделу је велики и износи $O\left(\binom{n}{k} \cdot n^2\right)$, а број ограничења је $O\left(\binom{n}{k} \cdot n\right)$, што значи да величина модела експоненцијално расте са величином инстанце. Стога су у случају већих инстанци за саму конструкцију модела потребни изузетно велики рачунарски ресурси. Алтернативни приступ заснован на Бендеровој декомпозицији такође има значајно ограничење у примени на велике графове, јер алгоритам показује спору конвергенцију ка висококвалитетним решењима.

Поред тога, предложени приступи су тестирани искључиво на скупу случајно генерисаних инстанци са до 30 чворова и до $k = 5$. Ове димензије графова нису довољно велике за реалне сценарије, што указује на поменуте проблеме са скалабилношћу. Осим тога, случајни графови често нису погодни за моделовање стварних система.

У литератури постоји још један рад у коме је k-SRD проблем изучаван са теоријске стране [95]. Одређене су и доказане тачне вредности k-SRD бројева за познате класе графова као што су потпуни графови, путеви, точкови и потпуни бипартитни графови.

Дакле, проблем је решаван само егзактним методама, које, како је већ поменуто, нису погодне за инстанце већих димензија у случају овако тешких комбинаторних проблема. Због тога је природно размотрити употребу апроксимативних метода као што су метахеуристике.

4.3 Метода променљивих околина за проблем k-SRD

Метода променљивих околина се показала као ефикасна у поглављу 2, као и у решавању како других доминацијских проблема [63, 100, 45], тако и проблема у разним областима међу којима су и: распоређивање [104, 2], алокацијски проблеми [57], рутирање [65, 60], биоинформатика [24, 52]. Стога је природно дизајнирати и применити ову методу на проблем k-SRD.

4.3.1 Унапред дефинисани напади и концепт квазидопустивости

Пре детаљног описа основних компоненти алгоритма, као што су размдравање и локална претрага, неопходно је истаћи суштински изазов везан за проверу допустивости решења овог проблема. Наиме, број потенцијалних напада који се морају проверити да би се утврдило да ли је решење допустиво или не, расте експоненцијално са величином инстанце. Ради превазилажења овог проблема, уведена је идеја о унапред дефинисаним нападима и квазидопустивости. Укратко, у случају превеликог броја могућих напада, уместо провере сваког од њих, проверавају се само они напади који су унапред дефинисани. На овај начин је омогућено решавање великог броја инстанци у разумном времену.

Унапред дефинисани напади

Напади се деле у две групе:

- **Интензивни напади** (означени са *intenseAttacks*) – користе се у циљу строже и поузданије провере квазидопустивости решења.

- **Лакши напади** (означени са *lightweightAttacks*) – користе се, како ће бити појашњено у наставку, током локалне претраге, као релаксирани облик провере квазидопустивости.

Код инстанци мале до средње величине, могуће је испитати све нападе, па се тада и за интензивне, и за лакше нападе користи потпуна провера квазидопустивости, која тада постаје еквивалентна допустивости. Граница до које је могуће испитати све нападе дефинисана је параметром *combTakeAllBound*. У случају да је $\binom{n}{k} < combTakeAllBound$, користе се сви могући k-напади $\mathcal{P}$. У супротном, генерише се подскуп $\mathcal{P}' \subseteq \mathcal{P}$ на следећи начин. За сваки чвор v дефинише се скуп $N_l(v)$, који садржи све чворове на удаљености највише l од v. За дати скуп чворова s и параметар k, са $comb(s, k)$ се означава скуп свих k-комбинација чворова из s, под условом да је $|s| \geq k$. Интензивни напади се дефинишу као $\bigcup_{v \in V} comb(N_3(v), k)$. Вредност $l = 3$ је одабрана јер обезбеђује добар баланс између добре апроксимације допустивости и рачунске ефикасности. Уколико је број интензивних напада мањи или једнак вредности *combTakeAllBound*, исти скуп се користи и као скуп лакших напада. У супротном, лакши напади се формирају као $\bigcup_{v \in V} comb(N_1(v), k)$.

На пример, за граф приказан на слици 4.1, скуп лакших напада, односно $comb(N_1(v), k)$, био би једнак $\{\{v_1, v_2, v_3\}, \{v_1, v_2, v_5\}, \{v_2, v_3, v_5\}, \{v_3, v_2, v_4\}, \{v_1, v_3, v_4\}\}$. Напад $\{v_1, v_4, v_5\}$ не би био укључен у скуп лакших напада, јер не постоји чвор коме су сви они суседи. Са друге стране, овај напад би био део скупа интензивних напада, који би се у овом случају састојао од свих могућих комбинација од 3 чвора, што је укупно 10 различитих напада.

Квазидопустивост

На основу претходно дефинисаних напада, концепт квазидопустивости омогућава практичну проверу допустивости решења, уместо провере свих напада. На овај начин се егзактан експоненцијалан алгоритам испитивања допустивости мења полуодлучивим алгоритмом полиномске сложености. У овом контексту, полуодлучивим алгоритмом се сматра онај који ако врати негативан одговор, онда је решење недопустиво, али у случају позитивног одговора не гарантује допустивост решења.

Конкретно, за сваки напад из скупа интензивних напада, проверава се да ли се он може успешно одбранити у складу са дефиницијом проблема. Уколико се пронађе бар један напад који је неодбрањив, решење се сматра недопустивим. У супротном, решење је квазидопустиво. Провера одбрањивости неког конкретног напада приказана је у алгоритму 7.

За потребе заштите чворова унутар одабраног скупа напада развијене су две стратегије одбране: детерминистичка и стохастичка, дате у алгоритмима 15 и 16 (детаљи имплементације у прилогу B). Избор између ове две стратегије врши се на основу броја алтернативних бранилаца свих тренутно незаштићених чворова, као и параметра *cutoff*. Уколико је број алтернатива релативно мали, могуће је исцрпно испитати све комбинације потенцијалних бранилаца. У супротном, примењује се стохастички приступ заснован на рулетској селекцији, где чворови који имају већи потенцијал да успешно одбране други чвор имају већу вероватноћу да буду одабрани као браниоци. Пошто ова стратегија не гарантује успешну одбрану, поступак се понавља више пута, при чему се број покушаја контролише параметром *tries* > *0*.

Главна функција, названа `quasiInfeasibility` (алгоритам 8), за сваки напад $h \in intenseAttacks$ позива функцију `isAttackDefended` (алгоритам 7). Уколико се за дати напад не пронађе валидна одбрана, повећава се бројач неуспешно одбрањених напада. На крају, функција враћа укупан број таквих напада као индикатор степена недопустивости решења.

Алгоритам 7 Функција `isAttackDefended`

Улаз: решење s, $G = (V, E)$, k, *attack*, *cutoff* > 0, *tries* > 0
Излаз: пар *attackDefended*, *defendingNodes*

$allAlternatives, reducedAttack \leftarrow \emptyset$
for $v \in attack$ **do**
  if $s[v] > 0$ **then**
    $defendingNodes \leftarrow defendingNodes \cup v$
    continue
  end if
  $alternatives \leftarrow \{u \in N(v) \mid \texttt{isAvailable}(u)\}$ ▷ чворови који могу да бране v
  if $alternatives = \emptyset$ **then**
    return `False`, $\emptyset$
  end if
  $reducedAttack \leftarrow reducedAttack \cup v$
  $allAlternatives \leftarrow allAlternatives \times alternatives$
end for
if $|allAlternatives| < cutoff$ **then**
  return `deterministicDefense`$(allAlternatives, reducedAttack, defendingNodes)$
else
  return `rouletteDefense`$(reducedAttack, defendingNodes, tries)$
end if

Алгоритам 8 Функција `quasiInfeasibility`

Улаз: решење s, $G = (V, E)$, k, *attacks*, *cutoff* > 0, *tries* > 0
Излаз: пар *nonDefendedCount*, *coverageInfo*

$nonDefendedCount \leftarrow 0$
$coverageInfo \leftarrow \emptyset$
for $attack \in attacks$ **do**
  $defended, defendingNodes \leftarrow$ `isAttackDefended`$(s, G, k, attack, cutoff, tries)$
  if $defended$ **then**
    `updateCoverageInfo`$(defendingNodes, attack)$
  else
    $nonDefendedCount \leftarrow nonDefendedCount + 1$
  end if
end for

4.3.2 Репрезентација решења и иницијализација

Решење је представљено као листа ненегативних целих бројева дужине $|V|$, где се за сваки чвор чува број јединица које су постављене на њега. Да би се на ефикасан начина генерисало почетно решење, развијен је похлепни алгоритам заснован на принципу покривања чворова у графу. У наставку је дата детаљна анализа предложене методе, уз псеудокод у алгоритму 9.

На почетку, свим чворовима се додељује лабела 0, чиме се добија иницијално решење облика $s = [0, \ldots, 0]$. Такође, иницијализује се и скуп $covered = \emptyset$, који ће у току рада алгоритма чувати све чворове који су већ заштићени у односу на све могуће нападе.

У свакој итерацији се бира наредни чвор за лабелирање на основу похлепног критеријума $g(\cdot)$. Овај критеријум се за сваки још необрађени чвор рачуна као број његових суседа (укључујући и сам тај чвор) који још увек нису покривени, односно нису у скупу *covered*. Формално, за сваки чвор $v \in V \setminus covered$ вредност похлепне функције $g(v)$ се рачуна као:

$$g(v) = |N[v] \setminus covered|$$

Од свих таквих чворова, бира се онај са највећом вредношћу функције g, означен као v^*. Уколико постоји више чворова са истом максималном вредношћу, предност се даје

чворовима који још увек нису покривени. Овакав секундарни критеријум у случају нерешених резултата има следеће оправдање:

- Ако, на пример, чвор v има $g(v) = 3$ и није покривен, он треба да покрије самог себе и још два суседа, што значи да су довољне 3 јединице.
- Насупрот томе, ако је чвор v већ покривен и такође има $g(v) = 3$, тада мора да покрије три суседа, али и да сам буде заштићен, што укупно захтева 4 јединице.

Изабраном чвору v^* се додељује вредност $l_{v^*} = \min(k + 1, g(v^*) + I_{v^* \in covered})$, где је $I_{v^* \in covered}$ индикаторска функција која враћа 1 уколико је чвор v^* већ покривен, а 0 у супротном. Овом лабелом се ажурира решење $s[v^*] = l_{v^*}$, а сви суседи чвора v^*, као и сам тај чвор, додају се у скуп *covered*. Алгоритам се завршава када сви чворови постану покривени.

Алгоритам 9 Функција `greedy`

Улаз: инстанца I k-SRD проблема, k
Излаз: решење s

1: $s \leftarrow [0, \ldots, 0]$
2: $covered \leftarrow \emptyset$
3: **while** $covered \neq V$ **do**
4: $v^* \leftarrow \arg\max\{g(v) \mid v \in V \setminus covered\}$ ▷ у случају једнакости, предност имају непокривени чворови
5: $uncov_{v^*} \leftarrow N[v^*] \setminus covered$
6: **if** $v^* \in uncov_{v^*}$ **then**
7: $countSelf \leftarrow 0$
8: **else**
9: $countSelf \leftarrow 1$
10: **end if**
11: $s[v^*] \leftarrow \min(k + 1, g(v^*) + countSelf)$
12: $covered \leftarrow covered \cup uncov_{v^*}$
13: **end while**

4.3.3 Функција прилагођености

Функција прилагођености за дато решење s дефинисана је као уређени пар $fitness(s) = (\textit{quasi-infeasibility}(s), sum(s))$, где:

- $\textit{quasi-infeasibility}(s)$ представља број неодбрањивих напада за решење s,
- $sum(s)$ означава збир свих лабела у решењу s.

За потребе локалне претраге, имплементирана је убрзана процедура за израчунавање вредности функције прилагођености. При сваком позиву функције `quasiInfeasibility`, генерише се информација о покривености напада, која се чува у одговарајућој структури података под називом *coverageInfo*. Ова структура бележи за сваку одбрањени напад $h \in$ *lightweightAttacks* чвор v који је искоришћен за одбрану при том нападу. Захваљујући овом механизму, знатно се убрзава процес локалне претраге, јер омогућава ефикасну проверу да ли су неки од напада из скупа *lightweightAttacks* већ одбрањени за решења s' која се налазе у околини решења s.

Размотримо граф са слике 4.1. Вредност функције прилагођености решења $s = (1, 0, 2, 1, 1)$ износи $(0, 5)$. Наиме, у оквиру скупа *intenseAttacks*, при примени

детерминистичке стратегије за проверу одбране (видети одељак 4.3.1), може се утврдити да за сваки од 10 напада постоји валидна стратегија одбране у складу са условима проблема. Са друге стране, решење $s = (1, 0, 2, 1, 0)$ има вредност функције прилагођености $(6, 4)$. У овом случају, сваки напад који укључује чвор v_5 остаје неодбрањив, при чему постоји управо $6 = \binom{4}{2}$ таквих напада који се могу формирати унутар скупа *intenseAttacks*.

4.3.4 Размрдавање

У оквиру методе променљивих околина, ова фаза је задужена за диверзификацију, односно уношење разноврсности у процес претраге. Реализована је функцијом `shake`(s, r), где је s тренутно решење, а r индекс околине $\mathcal{N}_r$. Излаз ове функције је измењено решење s', добијено на следећи начин. За дато решење s, извршава се r случајних увећања за 1 и $r + 1$ случајних умањења за 1 елемената решења, при чему се умањења примењују само на позиције чије лабеле нису нула. Избор позиција на којима се врши умањење заснива се на рулетској селекцији постојећих лабела у решењу s, што значи да позиције са већим вредностима имају већу вероватноћу да буду изабране.

На пример, за решење $s = [2, 3, 4, 1]$ и вредност $r = 2$, након два случајна увећања добијамо, рецимо, $s' = [3, 3, 4, 2]$, а након три умањења можемо добити $s' = [1, 3, 3, 2]$. Дакле, $\omega(s) - \omega(s') = 1$, односно укупан број јединица у решењу се смањује за један.

4.3.5 Локална претрага

Основна намена локалне претраге је интензификација, односно проналазак квалитетнијег локалног оптимума у блиској околини тренутног решења. За проблем k-SRD осмишљена је ефективна локална претрага са првим побољшањем која је заснована на принципу квазизамене (енг. *quasi-swap*). Имајући у виду да процес локалне претраге може бити временски захтеван, користи се релаксирана провера допустивости, уз помоћ унапред дефинисаног скупа напада *lightweightAttacks*.

Како је у фази размрдавања смањен укупан број јединица у решењу, највероватније, осим у случају велике редундантности, добијено решење је недопустиво. Циљ локалне претраге је да без промене укупног броја јединица у решењу, пронађе ново квазидопустиво решење.

Оператор квазизамене ради на паровима чворова, а с обзиром на то да се прихвата прво побољшање, потребно је на почетку сваке итерације конструисати насумичну пермутацију скупа парова чворова (i, j), при чему је $i < j$, ради уклањања евентуалне пристрасности у редоследу. Након тога, за сваки пар (i, j), разматрају се све могуће 2-декомпозиције збира $s[i] + s[j]$. На пример, ако је $s[i] = 2$, а $s[j] = 3$, разматра се 2-декомпозиција броја 5, што даје скуп: $\{(0, 5), (1, 4), (3, 2), (4, 1), (5, 0)\}$. Затим се, за сваку 2-декомпозицију, формира кандидатско решење заменом вредности $s[i]$ и $s[j]$ компонентама дате декомпозиције, нпр. $\{(s[i] = 0, s[j] = 5), (s[i] = 1, s[j] = 4), \ldots\}$.

Проверава се квазидопустивост сваког кандидата, тако што се израчунава број неодбрањивих напада. Уколико је број неодбрањивих напада мањи од броја неодбрањивих напада у тренутном решењу, тада се ново решење прихвата. У супротном се одбацује и прелази се на наредну 2-декомпозицију. У случају да ниједна 2-декомпозиција не даје боље решење, прелази се на следећи пар (i, j) и поступак се понавља.

Ако се ни за један пар (i, j) не пронађе боље решење, локална претрага се завршава и враћа се иницијално решење заједно са његовом вредношћу функције прилагођености. У противном, ако се током локалне претраге број неодбрањивих напада сведе на нулу, што указује на то да је пронађено ново (квази)допустиво решење, локална претрага се

завршава и враћа се ново решење са побољшаном вредношћу функције прилагођености.

4.3.6 Целокупна структура VNS алгоритма

На крају, у овом одељку је представљена целокупна структура предложеног VNS алгоритма за решавање k-SRD проблема, како је приказано у алгоритму 10.

Алгоритам 10 VNS за k-SRD проблем

Улаз: инстанца I k-SRD проблема, $k, r_{\min}, r_{\max}, move_{prob}, t_{\max}, iter_{max}, tries, cutoff$
Излаз: решење s_{best}

$intenseAttacks, lightweightAttacks \leftarrow$ `generateAttacks`(I, k)
$s_{best} \leftarrow$ `greedy`(I, k)
$infeasibility_{s_{best}} \leftarrow$ `quasiInfeasibility`$(s_{best}, intenseAttacks, cutoff, tries)$
$fitness_{s_{best}} \leftarrow (infeasibility_{s_{best}}, sum(s_{best}))$
while ! `terminationCriteriaSatisfied`() **do**
    for $r = r_{\min}$ *to* $r_{\max}$ **do**
        $s' \leftarrow$ `shake`(s_{best}, r)
        $s', fitness_{s'} \leftarrow$ `LS`$(s', lightweightAttacks, tries)$
        if $fitness_{s'} < fitness_{s_{best}} \vee (fitness_{s'} = fitness_{s_{best}} \wedge move_{prob} < u \in \mathcal{U}(0,1))$ **then**
            $intenseInfeasibility_{s'} \leftarrow$ `quasiInfeasibility`$(s', intenseAttacks, cutoff, tries)$
            if $intenseInfeasibility_{s'} = 0$ **then**
                $s_{best} \leftarrow s'$
                $fitness_{s_{best}} \leftarrow (intenseInfeasibility_{s'}, sum(s'))$
                break
            end if
        end if
    end for
end while

У првој линији алгоритма 10 генеришу се две врсте напада, описане у одељку 4.3.1. Након тога, иницијално решење се генерише похлепним алгоритмом из одељка 4.3.2. Потом се проверава квазидопустивост почетног решења уз помоћ функције `quasiInfeasibility` (видети одељак 4.3.1) и израчунава се вредност функције прилагођености дефинисане у одељку 4.3.3.

Након иницијализације, започиње главна петља VNS алгоритма, која се извршава све док не буде премашено дозвољено време или максималан број итерација. Основна итерација VNS алгоритма састоји се од следећих корака. Систематски се пролази кроз различите околине, почевши од најмање, означене са $r_{\min}$. У оквиру текуће околине r, позива се функција `shake` (видети одељак 4.3.4), која генерише ново решење s'. Ово решење се затим унапређује применом локалне претраге `LS` (видети одељак 4.3.5). Следи упоређивање тренутног најбољег решења s_{best} и новодобијеног решења s'. Уколико ново решење има бољу вредност функције прилагођености, односно број неодбрањивих напада у s' је мањи него у s_{best}, или ако је број једнак, али је укупан број јединица у s' мањи, тада се врши додатна верификација решења s'. Та верификација подразумева поновно извршавање функције `quasiInfeasibility` са пуним скупом $intenseAttacks$. Поред тога, ако су вредности функције прилагођености решења s' и s_{best} идентичне, примена верификације квазидопустивости за s' се врши са вероватноћом дефинисаном параметром $move_{prob}$.

Ако је након верификације број неодбрањивих напада у решењу s' једнак нули, оно се прихвата као ново најбоље решење. Наредна итерација VNS алгоритма тада почиње поново од најмање околине. У случају да се прође кроз све околине, а да ниједно ново решење не буде прихваћено, наредна итерација поново почиње од најмање околине.

4.4 Експериментални резултати

Предложени Vns алгоритам је упоређен са конкурентским приступима из литературе, што су методе засноване на целобројном линеарном програмирању и Бендеровој декомпозицији, означене са Ilp и Benders, описане у раду [84]. Поред тога, у поређење је укључен и похлепни алгоритам из одељка 4.3.2, означен као Greedy, са циљем процене релативног побољшања квалитета решења добијених применом комплетне Vns методе, у односу на иницијална решења генерисана похлепним алгоритмом.

Изворни код конкурентских метода Ilp и Benders је добијен од првог аутора рада [84]. Методе Vns и Greedy су имплементиране у програмском језику `C++17`, превођене су уз $Ofast$ оптимизациони ниво, а извршаване су на оперативном систему Ubuntu 20. Ради лакше употребе, развијен је Python модул `ksrdp`, који пружа интерфејс за рад са оригиналним `C++` кодом. Омогућена је једноставна инсталација путем следеће команде: `pip install ksrdp`. Поред тога, комплетан изворни код, скупови инстанци и сирови резултати експеримената слободно су доступни на GitHub репозиторијуму на адреси: `https://github.com/StefanKapunac/ksrdp`. Сви експерименти су изрвшавани у режиму са једном нити, на процесору Intel Xeon E5-2640 са радним тактом од 2.40GHz и 32 GB меморије.

4.4.1 Инстанце проблема

Експериментална евалуација је спроведена на три скупа референтних инстанци: Random, Wireless и Real. Скуп Random садржи насумично генерисане графове, првобитно преедложене у раду [84], где се могу наћи додатни детаљи. За сваку величину графа $n \in \{10, 15, 20, 30, 45, 50, 100\}$ генерисано је по 5 графова различитих густина, што укупно даје 35 инстанци.

Скуп Wireless се састоји од 16 инстанци које симулирају ад-хок бежичне мреже различитих густина. За генерисање ових графова коришћен је модел јединичног диска (енг. *Unit disc model*) [29]. У овом моделу постоје два параметра: полупречник r и број чворова n. Прво се генерише n насумичних тачака унутар јединичног квадрата у равни. Потом се сваки пар тачака на удаљености мањој од r повезује граном и добија се граф. Укупно је генерисано 16 инстанци, за вредности $n \in \{20, 30, 50, 100\}$ и $r \in \{0.3, 0.4, 0.5, 0.6\}$. За имплементацију је коришћена Python библиотека `NetworkX`, тачније функција `random_geometric_graph`.

Скуп Real се састоји од инстанци које су посебно осмишљене за потребе студије случаја стварне примене k-SRD проблема, представљене у одељку 4.5. За креирање ових графова искоришћена је колекција GeoJSON датотека доступних на GitHub репозиторијуму: `https://github.com/blackmad/neighborhoods.git`. Наведени скуп садржи 168 фајлова који кодирају географске карактеристике коришћењем структура као што су тачке, линије и полигони. На пример, могу представљати градове подељене на општине, државе подељене на регионе, или континенте подељене на државе. За обраду ових података коришћени су Python модули PySAL и GeoPandas, специјализовани за рад са геопросторним форматима као што је GeoJSON. Поступак генерисања графа из оваквог формата састоји се од два корака. Прво, за сваки полигон одређује се центроид који постаје чвор графа. Потом се конструише матрица суседства, у којој су два чвора повезана ако се полигони који им одговарају додирују, односно деле бар једну граничну тачку. Због тога се овакав граф, по угледу на кретање истоимене шаховске фигуре, назива краљичин (енг. *Queen's graph*) [53]. Илустрација једне овакве инстанце, која представља поделу града Берлина на општине, дата је на слици 4.3.

Слика 4.3: Граф општина у Берлину

4.4.2 Параметри алгоритма

На основу прелиминарних резултата, инстанце су категорисане у три групе:

- *мале* – инстанце са мање од 30 чворова,
- *средње* – инстанце са више од 30 и мање од 100 чворова,
- *велике* – остале инстанце.

За *мале* инстанце дозвољено време извршавања је 300 секунди, за *средње* инстанце лимит је продужен на 600 секунди, а за *велике* инстанце временско ограничење је 1200 секунди, односно 20 минута. Важно је напоменути да је исто временско ограничење примењено на сва четири поређена приступа. Поред тога, VNS има додатни критеријум заустављања, а то је број итерација који је ограничен на 5000 за све инстанце.

Следеће вредности параметара коришћене су у VNS алгоритму у свим експериментима: $r_{min} = 1$, $move_{prob} = 0.5$, $cutoff = 100$. Подешавање параметара r_{min} на најмању могућу вредност је уобичајена пракса у литератури [89, 56]. Слично је и са параметром $move_{prob}$, који је подешен на 0.5, односно одговара вероватноћи бацања новчића, што је такође честа стратегија у литератури. Вредност параметра *cutoff* одабрана је у складу са величином највећих инстанци. Не очекује се значајнији утицај овог параметра на перформансе алгоритма код великих графова, јер се у таквим случајевима користити пробабилистичка стратегија за проверу одбране од напада.

Параметри r_{max} и *tries* представљају најосетљивије вредности у оквиру алгоритма, јер директно утичу на делотворност кључних делова алгоритма, попут локалне претраге, размрдавања и евалуације функције прилагођености. Стога је извршена анализа утицаја ова два параметра на перформансе алгоритма која ће бити представљена у одељку 4.4.3. На основу ове анализе су изабране коначне вредности параметара VNS алгоритма чији су резултати укључени у поређење.

4.4.3 Резултати

У овом одељку су изложени експериментални резултати четири конкурентска приступа. Сви експерименти су изведени за $k \in \{2, 3, 4, 5\}$, уз примену исте методологије као у [84].

Табела 4.1: Анализа утицаја параметара алгоритма VNS: просечан квалитет решења на скупу RANDOM

$tries/r_{\max}$	5	10	50	100
5	**27.39**	28.49	27.89	27.89
10	**27.39**	27.52	27.88	27.89
50	27.40	27.52	27.87	27.88

Табела 4.2: Анализа утицаја параметара алгоритма VNS: просечан квалитет решења на скупу WIRELESS

$tries/\ r_{\max}$	5	10	50	100
5	12.12	12.16	12.17	12.16
10	**12.10**	12.15	12.16	12.17
50	12.12	12.13	12.15	12.15

Резултати на скупу инстанци RANDOM дати су у седам табела, груписаних према броју чворова, односно вредностима $n \in \{10, 15, 20, 30, 45, 50, 100\}$. Резултати на скупу WIRELESS представљени су у четири табеле, по једна за сваку вредност параметра k. Пре детаљне анализе добијених нумеричких резултата, дата је анализа утицаја два најзначајнија параметра приступа VNS – r_{max} и *tries*.

Анализа утицаја параметара

Анализа је изведена над следећим доменима параметара: $r_{max} \in \{5, 10, 50, 100\}$ и $tries \in \{5, 10, 50\}$. Свака комбинација ових вредности дефинише једну конфигурацију приступа VNS, што укупно даје 12 различитих конфигурација. Анализа је спроведена одвојено за сваки од два скупа инстанци – RANDOM и WIRELESS. Сумарни резултати, који представљају просечне вредности квалитета решења на свим инстанцама унутар сваког од скупова, дати су у табелама 4.1 и 4.2.

На основу агрегираних резултата добијених за 12 различитих конфигурација на оба скупа, изводе се следећи закључци:

- На скупу RANDOM, најбољи резултати постижу се при конфигурацијама $tries = 5$ и $r_{\max} = 5$, односно $tries = 10$ и $r_{\max} = 5$. Уочава се да повећање вредности параметра $r_{\max}$ доводи до погоршања резултата. На пример, значајна је разлика између резултата за $r_{\max} = 5$ и $r_{\max} = 50$. Са друге стране, параметар *tries* не утиче значајно на перформансе.

- На скупу WIRELESS, најбоља конфигурација је $tries = 10$ и $r_{\max} = 5$. Повећање параметра $r_{\max}$ доводи до благог смањења квалитета решења, али ове разлике нису статистички значајне. Као и у претходном случају, параметар *tries* показује мањи утицај у односу на $r_{\max}$.

Укратко, за финалну конфигурацију алгоритма VNS изабрана је вредност $r_{\max} = 5$ и $tries = 10$ за оба скупа инстанци. Треба напоменути да је фино подешавање параметара могло бити спроведено и уз помоћ алата за аутоматску конфигурацију, као што је Irace [85], чиме би можда били постигнути још бољи резултати. Међутим, оваква врста анализе је изостављена због ограничених рачунарских ресурса и већ спроведене анализе, која је указала на стабилну и поуздану конфигурацију.

Резултати на скупу Random

Резултати експеримената на скупу инстанци Random за инстанце код којих је $n \in \{30, 50, 100\}$ приказани су у табелама 4.3–4.5. Остали резултати дати су у прилогу Г, у табелама Г.1–Г.4. Све табеле су организоване на исти начин. Прве две колоне приказују карактеристике инстанце, односно број чворова n и густину *index*. У трећој колони се налази вредност параметра k. Следећа четири блока садрже резултате приступа Greedy, Vns, Ilp и Benders. За сваки од два егзактна приступа, дате су две статистике: квалитет добијеног решења (*obj*) и време извршавања у секундама (*t[s]*). Код Benders приступа додатно је приказана вредност доњег (дуалног) ограничења (*dual*). Време извршавања за Greedy алгоритам није приказано, јер је у свим случајевима било мање од једне секунде, што је значајно брже у поређењу са осталим приступима. За Vns, поред просечног квалитета решења и просечног времена проналаска најбољег решења у 10 извршавања, приказана је и релативна стандардна девијација (у процентима) квалитета добијених решења ($\sigma[\%]$). Најбољи резултати међу четири приступа су подебљани.

Из добијених нумеричких резултата (табеле 4.3–4.5) могу се извући следећи закључци:

- За инстанце са $n = 30$ и малим $k \in \{2, 3\}$, Ilp је успео да пронађе оптимална решења за све инстанце. Време извршавања драстично расте при преласку са $k = 2$ на $k = 3$. Benders приступ је у два случаја премашио ограничење у времену, али је дао дуалне границе које одговарају вредностима оптималних решења. Предложени Vns је дао решења квалитета једнаког оптималним, а у већини случајева имао је краће време извршавања од Ilp-а. Почетна решења добијена помоћу Greedy алгоритма значајно су побољшана током претраживања унутар Vns-а.

- За инстанце са $n = 30$ и већим $k \in \{4, 5\}$ није било могуће ни конструисати одговарајући ILP модел услед ограничења меморије. Benders приступ није успео да нађе доказано оптимално решење ни за један пример. Просечни квалитет решења Vns приступа био је значајно бољи од оног код Greedy приступа. Доње границе добијене Benders методом указују да решења Vns-а нису далеко од оптималних, а понекад и достижу оптимум.

- Код инстанци са $n = 50$ и $k = 2$, Ilp је постигао оптимална решења на свих пет инстанци, док је Benders то успео на три, а на преостале две је прекорачено време извршавања. Просечни квалитет решења Vns приступа био је једнак оптималним на три од пед инстанци. Vns је на свим инстанцама значајно побољшао почетна решења у односу на Greedy.

- За инстанце са $n = 50$ и $k \in \{3, 4, 5\}$, егзактни приступи Ilp и Benders нису успели да пронађу оптимално решење услед проблема са меморијским или временским ограничењима. Vns је поново дао значајно боља просечна решења од Greedy алгоритма.

- На инстанцама са $n = 100$ и $k = 2$, Ilp је пронашао оптимална решења у свих пет случајева, док Benders није постигао ниједно, при чему су дуалне границе биле релативно слабе. Vns је дао решења која просечно одступају од оптимума око 4%. Опет, Vns значајно побољшава почетна решења добијена Greedy методом.

- Код инстанци са $n = 100$ и $k \in \{3, 4, 5\}$, Ilp није могао да пронађе оптимална решења због проблема са меморијом. Benders је пробио временско ограничење и дао слабе дуалне границе. Vns је поново дао знатно боља просечна решења од Greedy метода – у неким случајевима релативна побољшања достижу и 10%.

Табела 4.3: Упоредни резултати на скупу RANDOM за $n = 30$.

инстанца		k	GREEDY	VNS			ILP		BENDERS		
n	*index*		obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
30	1	2	20	**17.0**	0.00	2.0	**17**	4.7	**17**	17	41.2
	2		20	**17.0**	0.00	16.1	**17**	6.5	**17**	17	28.4
	3		21	**17.0**	2.8	2.2	**17**	8.3	**17**	17	87.4
	4		17	**16.0**	0.0	0.1	**16**	1.1	**16**	16	36.0
	5		18	**17.0**	8.0	12.63	**17**	6.6	**17**	17	23.3
30	1	3	23	**19.0**	0.0	44.1	**19**	208.9	0	19	TL
	2		23	**20.0**	0.0	21.0	**20**	222.2	**20**	20	373.6
	3		25	**20.0**	0.0	8.9	**20**	380.5	0	19	TL
	4		21	**19.0**	0.0	3.7	**19**	185.2	**19**	19	423.9
	5		22	**20.0**	0.0	21.7	**20**	19.6	**20**	20	491.9
30	1	4	26	**20.6**	2.0	226.0	-	-	0	19	TL
	2		26	**21.0**	0.0	70.1	-	-	0	20	TL
	3		29	**21.0**	0.0	148.6	-	-	0	19	TL
	4		24	**20.2**	1.6	142.6	-	-	0	19	TL
	5		25	**21.5**	2.2	139.1	-	-	0	20	TL
30	1	5	28	**22.1**	2.3	134.6	-	-	0	19	TL
	2		29	**22.6**	4.1	165.4	-	-	0	20	TL
	3		30	**23.4**	2.7	194.4	-	-	0	19	TL
	4		26	**21.1**	3.2	249.8	-	-	0	19	TL
	5		27	**23.0**	2.0	164.6	-	-	0	20	TL

Табела 4.4: Упоредни резултати на скупу RANDOM за $n = 50$.

инстанца		k	GREEDY	VNS			ILP		BENDERS		
n	*index*		obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
50	1	2	31	**29.0**	0.0	15.0	**29**	32.9	0	28	TL
	2		31	**28.0**	0.0	7.0	**28**	15.2	**28**	28	439.6
	3		34	28.2	0.0	116.1	**28**	28.7	**28**	28	399.8
	4		33	**28.0**	0.0	5.6	**28**	21.6	**28**	28	375.8
	5		36	29.2	0.0	137.8	**29**	25.2	0	28	TL
50	1	3	36	**33.4**	1.6	253.3	-	-	0	28	TL
	2		36	**32.4**	1.5	164.0	-	-	0	28	TL
	3		40	**32.2**	1.2	377.3	-	-	0	28	TL
	4		38	**32.7**	2.2	311.1	-	-	0	28	TL
	5		43	**34.7**	3.0	238.5	-	-	0	29	TL
50	1	4	40	**36.0**	2.0	107.4	-	-	0	28	TL
	2		40	**35.1**	2.0	262.6	-	-	0	28	TL
	3		44	**36.9**	2.4	236.4	-	-	0	28	TL
	4		42	**35.0**	1.5	213.4	-	-	0	28	TL
	5		47	**37.6**	2.5	242.1	-	-	0	28	TL
50	1	5	44	**40.4**	2.9	257.7	-	-	0	28	TL
	2		44	**39.6**	2.0	307.4	-	-	0	28	TL
	3		48	**41.5**	2.4	429.2	-	-	0	27	TL
	4		45	**40.0**	2.5	378.6	-	-	0	28	TL
	5		50	**43.4**	2.9	337.2	-	-	0	29	TL

Табела 4.5: Упоредни резултати на скупу RANDOM за $n = 100$.

инстанца		k	GREEDY	VNS			ILP		BENDERS		
n	*index*		obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
100	1	2	65	58.9	313.6	505.1	**57**	402.6	0	49	TL
	2		68	61.2	1.2	461.2	**58**	752.2	0	49	TL
	3		66	62.2	0.9	460.0	**60**	775.1	0	52	TL
	4		64	60.4	2.1	492.1	**58**	296.5	0	50	TL
	5		61	59.0	0.5	189.6	**57**	315.7	0	50	TL
100	1	3	79	**73.5**	2.8	964.8	-	-	0	49	TL
	2		82	**75.6**	2.4	915.2	-	-	0	49	TL
	3		79	**76.3**	1.1	441.7	-	-	0	51	TL
	4		77	**75.5**	1.1	600.2	-	-	0	50	TL
	5		72	**69.5**	1.0	681.1	-	-	0	49	TL
100	1	4	86	**77.4**	3.0	898.8	-	-	0	49	TL
	2		92	**81.6**	1.4	878.0	-	-	0	49	TL
	3		89	**81.8**	1.8	822.4	-	-	0	51	TL
	4		86	**79.0**	2.2	573.4	-	-	0	50	TL
	5		81	**75.0**	2.0	603.8	-	-	0	50	TL
100	1	5	92	**89.4**	2.2	864.4	-	-	0	49	TL
	2		98	**90.1**	1.4	1050.8	-	-	0	49	TL
	3		95	**91.8**	1.3	936.2	-	-	0	51	TL
	4		94	**90.7**	1.6	1020.3	-	-	0	49	TL
	5		88	**85.9**	1.4	870.7	-	-	0	49	TL

Резултати на скупу WIRELESS

Експериментални резултати на скупу WIRELESS груписани су према различитим вредностима параметра k и приказани у четири одвојене табеле (по једна табела за сваку вредност k). Резултати за инстанце са $k \in \{3, 4\}$ приказани су у табелама 4.6–4.7, док се остали резултати могу наћи у табелама Г.5–Г.6 у прилогу Г. Табеле су организоване слично као у претходној секцији, осим колона које приказују карактеристике инстанце. Прва колона приказује величину графа (n), док друга колона приказује вредност радијуса (R), који одговара густини графа (што је већи радијус, већа је густина графа). Поново су најбољи резултати међу четири поређена приступа приказани подебљано.

Из ових резултата могу се извући следећи закључци:

- За инстанце са $n = 20$ и $k = 3$, ILP и BENDERS приступи су ефикасно пронашли оптимална решења на свим инстанцама. VNS је постигао оптимална решења уз изузетно кратко време извршавања. Резултати GREEDY приступа не одступају много од оптималних решења. Приметно је да се инстанце веће густине лакше решавају.

- На већим инстанцама са $n \geq 50$ и $k = 3$, егзактни приступи нису успели да пронађу оптимална решења, услед меморијских и временских ограничења. Изузетак је једна инстанцу где је BENDERS био успешан, али уз дуже време извршавања. У седам од осам случајева, решења VNS приступа су значајно боља од решења добијених GREEDY методом.

- За најмање инстанце са $n = 20$ и $k = 4$, егзактни ILP приступ је имао потешкоће и успео је да реши само једну инстанцу ($n = 20$ и $R = 0.3$). BENDERS показује боље перформансе, решивши три од четири инстанце. VNS је достигао оптимална решења за све ове инстанце у оквиру једног минута.

- За веће инстанце са $n \geq 50$ и $k = 4$, егзактни ILP и BENDERS приступи нису успели да реше ниједну инстанцу због проблема са меморијским и временским ограничењима.

Табела 4.6: Упоредни резултати на скупу WIRELESS за $k = 3$.

инстанца		GREEDY	VNS			ILP		BENDERS		
n	R	obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
20	0.3	15	**14.0**	0.0	0.2	**14**	5.73	**14**	14	31.1
	0.4	**12**	**12.0**	0.0	0.0	**12**	51.11	**12**	12	101.0
	0.5	7	**6.0**	0.0	1.7	**6**	41.56	**6**	6	44.6
	0.6	**4**	**4.0**	0.0	0.0	**4**	3.14	**4**	4	9.8
50	0.3	18	**16.0**	0.0	205.4	-	-	0	15	TL
	0.4	14	**13.1**	0.0	106.9	-	-	0	11	TL
	0.5	10	**8.3**	0.0	173.5	-	-	0	7	TL
	0.6	7	**6.0**	0.0	90.6	-	-	**6**	6	1402.7
100	0.3	23	**21.3**	10.1	461.6	-	-	0	13	TL
	0.4	17	**14.3**	8.3	393.5	-	-	0	10	TL
	0.5	**12**	**12.0**	0.0	0.0	-	-	0	7	TL
	0.6	9	**8.2**	5.1	222.7	-	-	0	6	TL

Табела 4.7: Упоредни резултати на скупу WIRELESS за $k = 4$.

инстанца		GREEDY	VNS			ILP		BENDERS		
n	R	obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
20	0.3	17	**16.0**	0.0	1.9	**16**	135.2	**16**	16	144.3
	0.4	14	**13.0**	0.0	44.2	19	TL	0	13	TL
	0.5	8	**7.0**	0.0	13.1	43	TL	**7**	7	267.9
	0.6	**5**	**5.0**	0.0	0.0	46	TL	**5**	5	33.1
50	0.3	**21**	**21.0**	2.8	0.0	-	-	0	15	TL
	0.4	16	**15.4**	0.0	78.1	-	-	0	12	TL
	0.5	12	**11.5**	0.0	67.9	-	-	0	8	TL
	0.6	8	**7.0**	0.0	17.5	-	-	0	7	TL
100	0.3	28	**27.5**	0.0	352.0	-	-	0	14	TL
	0.4	21	**20.3**	3.3	210.2	-	-	0	10	TL
	0.5	**15**	**15.0**	5.1	0.0	-	-	0	8	TL
	0.6	11	**10.7**	0.0	218.3	-	-	0	7	TL

VNS је постигао боља решења у шест од осам инстанци у поређењу са резултатима GREEDY алгоритма.

Сумарни резултати на скупу REAL

Ради провере ефективности предложеног VNS приступа на реалним инстанцама из скупа REAL, примењена је следећа методологија. Узете су у обзир оне инстанце за које оба егзактна приступа могу обезбедити оптимална решења. Затим се квалитет добијених решења GREEDY и VNS приступа упоређује са оптималним решењима, груписаних по вредностима $k \in \{2, 3, 4, 5\}$ (по један ред за сваку вредност k). Ови резултати приказани су у табели 4.8. Због великог броја инстанци VNS је извршаван само једном по инстанци. С обзиром на релативно малу стандардну девијацију добијену за решења VNS-а на скуповима RANDOM и WIRELESS, не очекује се да закључци добијени у сценарију са једним извршавањем буду битно другачији од оних који би били добијени са десет извршавања.

Из наведених резултата може се закључити следеће:

- Број инстанци које се могу решити оптимално брзо опада са повећањем вредности k.
- VNS проналази оптимална решења у свим случајевима где ILP или BENDERS решавају инстанцу (са неколико изузетака за случај $k = 2$).

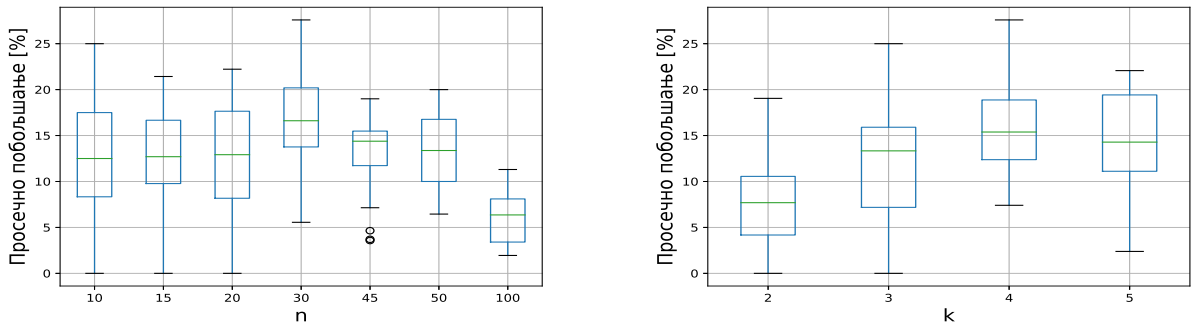
Табела 4.8: Поређење на скупу Real – случајеви које могу решити Ilp и Benders

k	Greedy	Vns	Ilp/Benders	број инстанци
2	20.98	20.18	**20.11**	66
3	15.58	**14.74**	**14.74**	43
4	13.96	**13.00**	**13.00**	25
5	12.47	**11.59**	**11.59**	17

- Greedy је значајно надмашен од стране Vns-а.
- Важно је нагласити да оба предложена хеуристичка приступа пружају апроксимативна решења разумног квалитета (значајно боља од тривијалног решења) и за оне инстанце где егзактне методе могу обезбедити било какво решење.

4.4.4 Поређење хеуристичких приступа

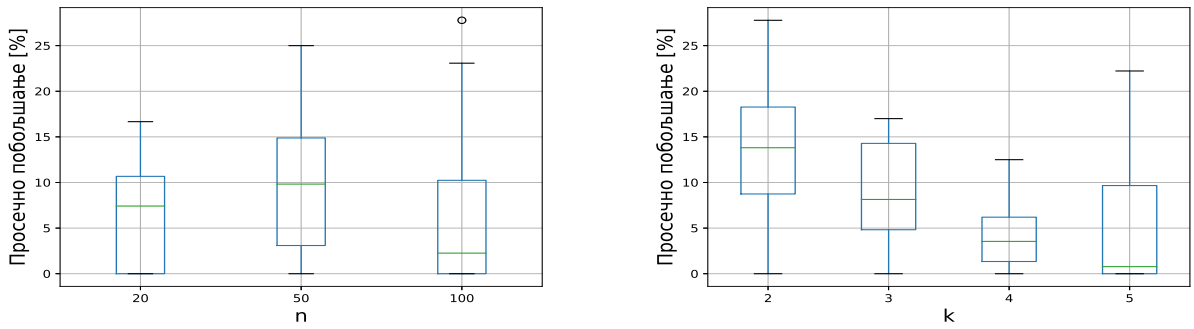
Да би се додатно упоредиле перформансе два хеуристичка приступа, Greedy и Vns, дате су слике 4.4а–4.4б за скуп инстанци Random и слике 4.5а–4.5б за скуп Wireless. Ови графици приказују просечно релативно побољшање квалитета решења добијених Vns приступом у односу на Greedy. Инстанце су груписане по величини графа n и по вредности k (вредности приказане на x-оси) посебно, те су по два графика генерисана за сваки скуп. Додатно, слика 4.6 приказује просечно релативно побољшање квалитета решења добијених Vns-ом у односу на Greedy за све инстанце скупа Real, при чему су инстанце груписане по вредности k.



(а) Инстанце груписане по n

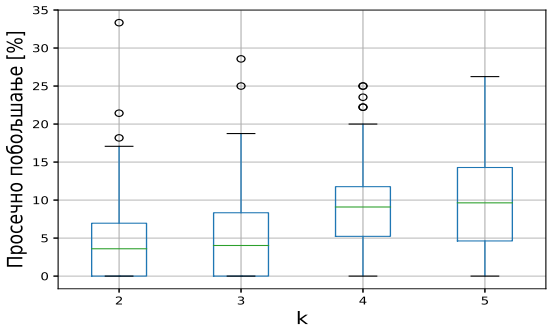
(б) Инстанце груписане по k

Слика 4.4: Релативна просечна побољшања Vns-а у односу на Greedy на скупу Random



(а) Инстанце груписане по n (б) Инстанце груписане по k

Слика 4.5: Релативна просечна побољшања VNS-а у односу на GREEDY на скупу WIRELESS



Слика 4.6: Релативна просечна побољшања VNS-а у односу на GREEDY на скупу REAL

Следећи закључци се могу извући из добијених графика:

- За инстанце из скупа RANDOM груписане по величини графа n, просечна релативна побољшања квалитета решења VNS-а у односу на GREEDY крећу се од респектабилних 12–16% за $10 \leq n \leq 50$, док је за групу највећих инстанци са $n = 100$ ова вредност и даље остаје значајна, око 5%.

- За инстанце из скупа RANDOM груписане по различитим вредностима k, просечна релативна побољшања VNS-а крећу се од 8.5% за $k = 2$ до око 14% за $k = 4$.

- За инстанце из скупа WIRELESS груписане по величини графа n, просечна побољшања VNS-а крећу се око 6% за највећи $n = 100$ и приближно 7.7% за мање $n \in \{20, 50\}$.

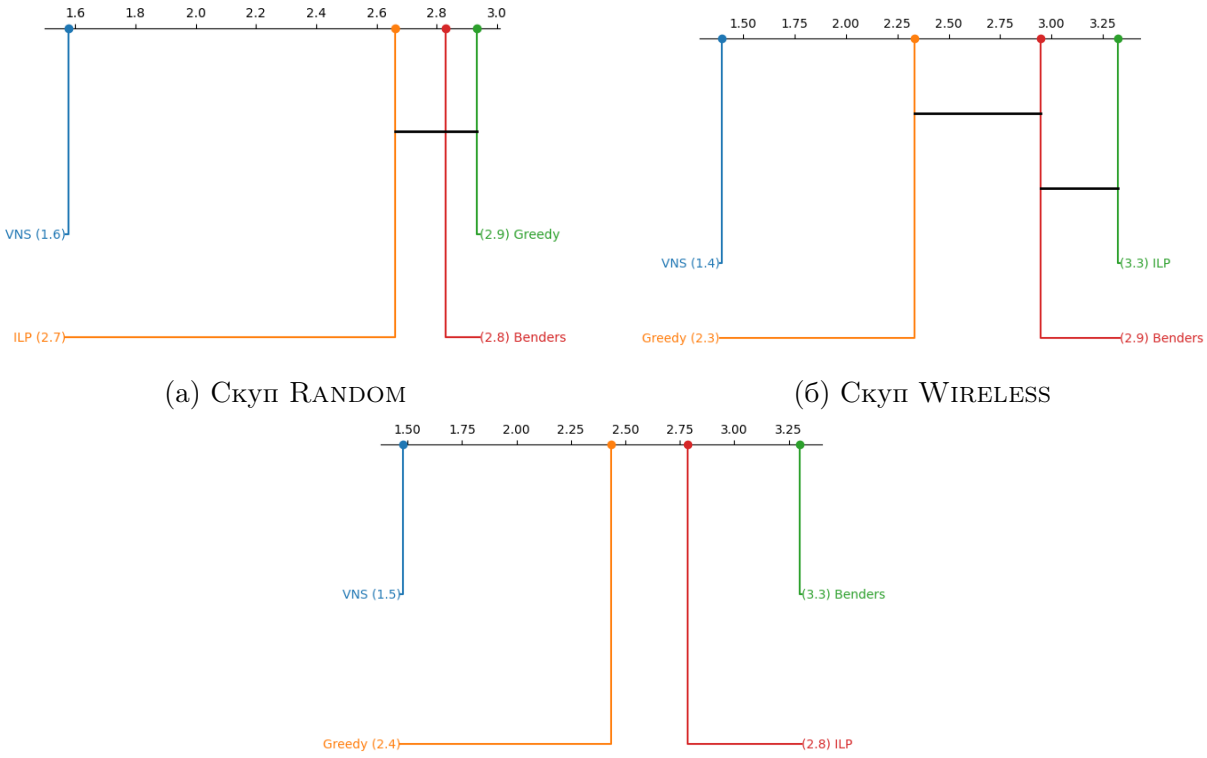
- За инстанце из скупа WIRELESS груписане по различитим вредностима k, просечна побољшања, очекивано, слабе са повећањем вредности k. За најмање $k = 2$ износе око одличних 15%, док се за највећи случај ($k = 5$) смањују на 3%. То је углавном последица великог броја генерисаних напада који морају бити проверени у смислу постојања стратегије одбране, што значајно смањује укупан број извршених итерација VNS-а.

- За инстанце из скупа REAL, просечна релативна побољшања VNS-а крећу се од приближно 4% за $k \in \{2, 3\}$ до око 10% за највећи случај $k = 5$.

4.4.5 Статистичка анализа

Ради провере статистичке значајности уочених разлика између четири приступа, примењена је следећа статистичка методологија. Прво, за све четири конкурентске методе је извршен Фридманов тест [47] на свим инстанцама проблема из скупова Random и Wireless, посматрано одвојено. Након тога, у случају одбацивања нулте хипотезе (H_0 тврди да су све конкурентске методе статистички подједнако добре), спроведена су упоредна испитивања у паровима коришћењем Неменјијевог пост-хок теста [93].

Резултати добијени овом анализом приказани су коришћењем графика критичне разлике (енг. *critical difference*, CD). Као што је раније поменуто, на овим графицима сваки приступ је позициониран на хоризонталној оси у складу са просечним рангом. Затим се за ниво значајности 0.05 израчунава критична разлика (CD). Уколико је разлика довољно мала, односно није откривена статистички значајна разлика, црта се хоризонтална линија која повезује статистички једнаке приступе. У случају да алгоритам није дао резултат, додељујемо велику вредност (нпр. 1000) како би се омогућио највећи могући ранг.



(а) Скуп Random

(б) Скуп Wireless

(в) Скуп Real

Слика 4.7: Графици критичне разлике – статистичке разлике међу 4 поређена приступа

Следећи закључци се могу извући из спроведене статистичке анализе, приказане на сликама 4.7а–4.7в:

- За скуп инстанци Random, Vns постиже најбољи просечан ранг у смислу квалитета решења. Остала три приступа су значајно иза, са просечним рангом близу три. Поред тога, Vns је статистички значајно бољи од остала три конкурента. Међу ова три приступа није откривена статистички значајна разлика.
- За скуп инстанци Wireless, Vns поново осигурава најбољи просечни ранг, који је за целу јединицу бољи од другопласираног приступа Greedy. Најгори просечни ранг имају Benders и Ilp, редом. Опет, најбоље рангирани приступ Vns статистички

значајно надмашује Greedy, који пак има статистички значајно бољи ранг од Ilp. Упркос мањем просечном рангу Greedy, није откривена статистички значајна разлика између Greedy и Benders. Приступи Benders и Ilp нису статистички различити, нити су могли да доставе било какво решење за веће вредности k.

- За скуп инстанци Real, Vns поново постиже најбољи просечни ранг. Други најбољи просечни ранг остварује приступ Greedy, углавном због конзистентног достављања допустивих решења за све инстанце, за разлику од два егзактна конкурента. Треба напоменути да су решења Vns-а статистички значајно боља од решења Greedy методе.

4.4.6 Анализа утицаја појединачних компоненти алгоритма

Спроведена је анализа утицаја појединачних компоненти Vns алгоритма на квалитет добијених решења. Конкретно, испитано је следеће:

- Утицај квалитета почетног решења које се прослеђује Vns-у. Проверена је осетљивост на почетно решење добијено Greedy методом у односу на тривијално почетно решење. Ова верзија Vns-а која користи тривијално решење као иницијално назива се Vns-trivial-init.

- Утицај Ls методе. Верзија Vns-а без Ls назива се редуковани Vns и означава се као Rvns.

Експериментална методологија коришћена за ове експерименте прати методологију из претходних секција. Сумирани резултати су приказани у табелама 4.9–4.10. Табела 4.9 приказује резултате за скуп инстанци Random, груписане по различитим вредностима n (редови). Следећи закључци се могу извући из тих резултата:

- За мале инстанце са $n \in \{10, 15\}$, квалитет почетног решења не игра значајну улогу за Vns. Кључна разлика овде је присуство локалне претраге, која се показује као веома ефективна и неопходна за мање инстанце.

- За средње инстанце са $n \in \{20, 30\}$, и квалитет почетног решења и коришћење локалне претраге делују подједнако важно за укупну ефикасност предложеног приступа.

- За велике инстанце са $n \geq 45$, утицај почетног решења, очекивано, постаје још значајнији, јер процедура локалне претраге захтева много више итерација да би се побољшало тривијално решење.

Табела 4.10 приказује резултате за скуп инстанци Wireless, груписане по различитим вредностима k (редови). Из тих резултата могу се извући следећи закључци:

- За случај $k \in \{2, 3\}$, улога локалне претраге је важна, заједно са добрим почетним решењем које се прослеђује главној петљи Vns-а.

- За случај $k \in \{4, 5\}$, улога добрих почетних решења постаје још значајнија. Међутим, без коришћења локалне претраге, односно коришћењем Rvns, није могуће достићи најбоље резултате добијене са потпуним Vns-ом. Ово показује ефикасност ових двеју компоненти у комбинацији.

Табела 4.9: Анализа утицаја појединачних компоненти алгоритма VNS на скупу RANDOM (агрегирано по n)

n	VNS	VNS-TRIVIAL-INIT	RVNS
10	**6.85**	**6.85**	7.01
15	**10.15**	**10.15**	10.44
20	**13.17**	13.18	13.79
30	**19.92**	20.21	21.05
45	**31.23**	32.16	33.11
50	**34.67**	36.13	36.79
100	**75.74**	80.91	77.05

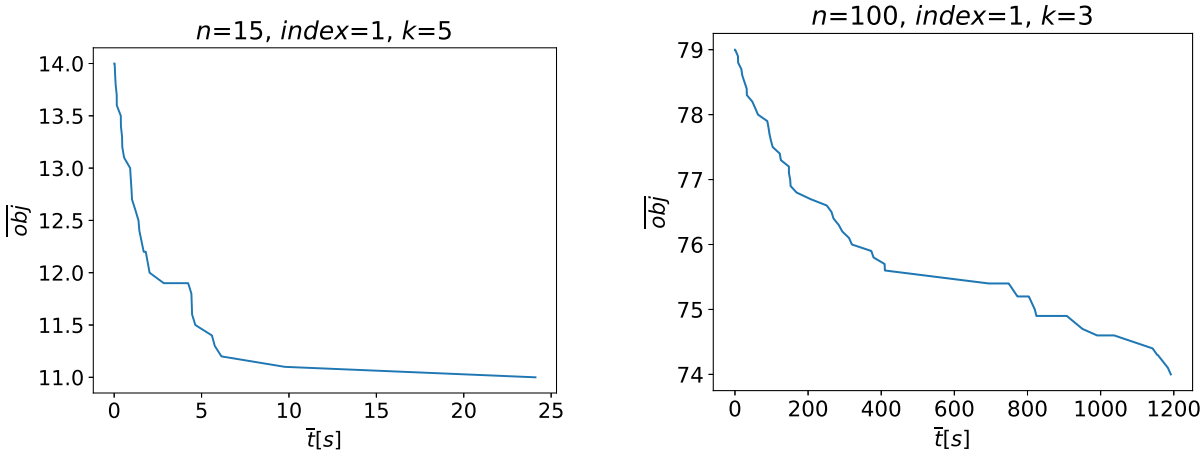
Табела 4.10: Анализа утицаја појединачних компоненти алгоритма VNS на скупу WIRELESS (агрегирано по k)

k	VNS	VNS-TRIVIAL-INIT	RVNS
2	**8.42**	**8.42**	9.76
3	**11.28**	11.53	12.03
4	**14.12**	31.51	14.27
5	**16.33**	46.10	16.56

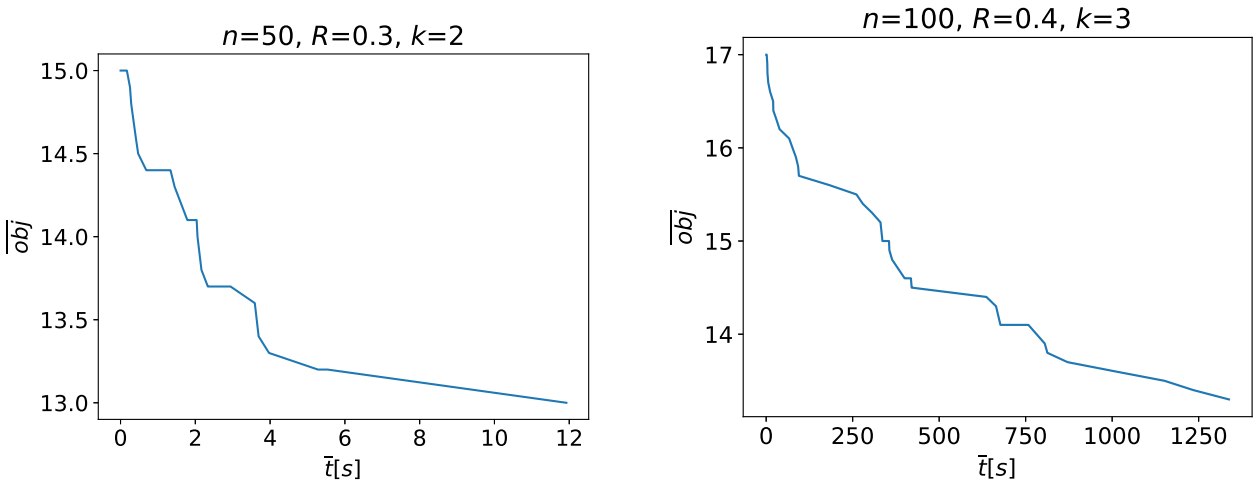
4.4.7 Анализа конвергенције алгоритма

Коначно, анализирана је конвергенција VNS алгоритма. На сликама 4.8–4.9 приказан је просечан профил перформанси алгоритма, где x-оса представља време (у секундама), а y-оса приказује квалитет решења који алгоритам постиже у сваком тренутку. Резултати су упросечени на основу 10 извршавања алгоритма по инстанци.

Треба напоменути да графици са леве стране представљају инстанце за које су достигнута позната оптимална решења. За графиконе са десне стране, VNS је завршен након достизања временског ограничења. Може се уочити да VNS доследно побољшава решења током времена, при чему су побољшања примећена и у каснијој фази претраге, након 1000 секунди. Ово додатно потврђује делотворност предложеног приступа.



Слика 4.8: Перформансе на одабраним инстанцама из скупа RANDOM



Слика 4.9: Перформансе на одабраним инстанцама из скупа Wireless

4.5 Примена у одбрани од више истовремених пожара

Да би се додатно истакла практична применљивост предложене методе, решења Vns-а су визуелизована и анализирана на неколико реалних инстанци из скупа Real, представљеног у секцији 4.4. Као студија случаја разматран је локацијски проблем којим се моделује постављање ватрогасних станица са возилима унутар града, при одређеним ограничењима.

Полазна основа за анализу заснива се на следећим статистичким подацима, углавном преузетим са Википедије. На пример, број ватрогасних станица у Даблину је 14, у Берлину 35, у Чикагу 98, док у Њујорку постоји 254 станице. Број ватрогасних возила у Чикагу износи 61, а у Њујорку 143. У Берлину је 2019. године Ватрогасна служба примила 478281 хитни позив, што је највећи број међу свим ватрогасним службама у Немачкој. Приближно 83% интервенција односи се на хитну медицинску помоћ, 5% на техничку асистенцију и око 2% на гашење пожара.* Уз претпоставку да је у просеку потребно око 4 сата за једну комплетну интервенцију (од изласка возила из станице до његовог повратка и поновне спремности), може се очекивати да је број реалних пожара који могу истовремено настати у овом граду око $k = 5$.

Формулисан је проблем постављања ватрогасних станица под следећим (поједностављеним) ограничењима:

- У свакој општини може бити изграђена највише једна ватрогасна станица.
- На свакој изабраној локацији може се стационирати једно или више возила.
- Једно возило може гасити пожар у својој или суседној општини. Ово одражава реалистичну потребу за брзом реакцијом у хитним случајевима, где би, на пример, слање возила са једног краја града на супротан крај трошило драгоцено време.
- Најмање једно возило увек мора остати у својој општини, док се остала могу слати у суседне дистрикте.

Дато k представља процену броја општина у којима могу истовремено постојати пожари. Додатна поједностављујућа претпоставка је да је једно возило довољно за гашење

*Информација је преузета са `https://en.wikipedia.org/wiki/Berlin_Fire_Brigade`.

било ког пожара. Циљ је пронаћи оптималне локације за изградњу станица и број возила у њима тако да сваки сценарио са k истовремених пожара може бити успешно покривен.

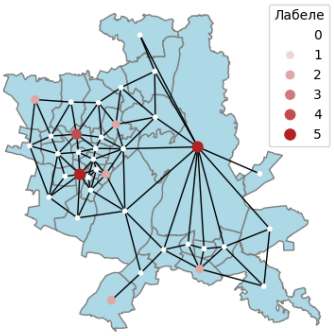
Визуелно је приказана примена овог проблема на неколико градова чије су графичке репрезентације добијене из GeoJSON фајлова (видети секцију 4.4). Изабрани су градови Шчећин и Синсинати као представници средње великих инстанци, и Питсбург као представник великих инстанци. Вредност k је узета из скупа $\{5, 6, 8\}$. Град Шчећин садржи 36 општина које формирају повезан граф. Град Синсинати обухвата 45 општина, такође у једном повезаном графу. Са друге стране, Питсбург садржи 90 општина подељених у четири повезане компоненте. Важно је истаћи да ниједан од егзактних приступа није успео да реши ове инстанце, док решења која даје VNS доследно надмашују она добијена GREEDY алгоритмом. Решења добијена применом VNS-а за сва три града (и све вредности k) приказана су на сликама 4.10а–4.12в. Следе закључци за град Шчећин:

- За $k = 5$ добијено решење износи 24. Изабрано је осам општина као чворишта за изградњу ватрогасних станица са најмање једним возилом. Две централизоване станице имају по пет возила, док једна има четири. Пет станица имају по два возила. Два највећа чворишта налазе се у географски највећој општини и у малој, али централној високо повезаној општини.

- За $k = 6$ решење износи 28. Планира се изградња девет станица. Једна станица има пет, једна шест, а једна седам возила. Остале имају по два или мање возила.

- За $k = 8$ добијено решење износи 33. Изграђено је осам станица. Три од њих доминирају бројем возила: две са по девет и једна са шест возила. Остале станице имају по два или мање возила.

- Занимљиво је да се три локације за станице са великим бројем возила појављују у сва три случаја $k \in \{5, 6, 8\}$. Ово се може објаснити њиховим високим степеном централности, што указује на добру доступност великог броја суседних општина. Три заједничка чвора за сваки број k имају степен централности 0.25, 0.31 и 0.2, што су вредностима међу највишима у целом графу.



(а) $k = 5$

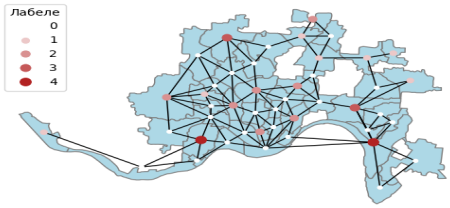
(б) $k = 6$

(в) $k = 8$

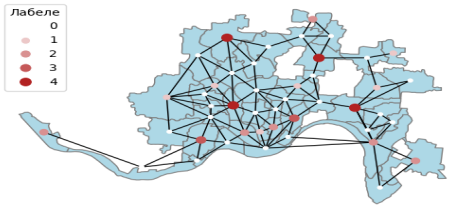
Слика 4.10: Резултати за град Шчећин

Следећи закључци су изведени за граф града Синсинатија:

- За $k = 5$, добијено решење износи 35. Предложено је отварање 18 ватрогасних станица. Ниједна станица нема више од четири возила. Постоје две станице са по четири возила и две са по три. Станице са већим бројем возила делују географски равномерно распоређене.
- За $k = 6$, добијено решење износи 40. Предложено је отварање 18 ватрогасних станица. Постоје три станице са по четири возила и две са по три, док преостале станице имају по два или мање возила.
- За $k = 8$, добијено решење износи 44. Предложено је отварање 17 ватрогасних станица. Једна станица има седам возила, три по пет, а две по четири возила, док је осам станица добило само по једно возило (углавном изолованији чворови који се налазе на границама градског подручја). Седам станица са највећим бројем возила делују географски равномерно распоређене по градском подручју.
- Чвор којем је додељен највећи број возила (7) има степен централности 0.16, што је друга највећа вредност у графу.



(а) $k = 5$



(б) $k = 6$

(в) $k = 8$

Слика 4.11: Резултати за град Синсинати

На крају, приказано је и анализирано решење добијено методом VNS за град Питсбург. Примећено је следеће:

- За $k = 5$, добијено решење износи 67. Предложено је 27 локација за изградњу ватрогасних станица. Две станице имају по пет возила, четири по четири возила, а шест по три возила. Преостале станице имају два или мање возила.
- За $k = 6$, добијено решење износи 72. Одабрано је 30 локација за изградњу ватрогасних станица. Три станице имају по пет возила, три по четири, а шест по три возила.
- За $k = 8$, добијено решење износи 81. Предложено је 27 локација за изградњу ватрогасних станица. Четири станице имају по шест возила, две по пет, а четири по три возила.
- За свако k, локације станица са највећим бројем возила у великој мери се поклапају. Такође, две локације са већим бројем возила имају међу највишим степенима централности у графу, једна 0.09, а друга 0.06.

(а) $k = 5$

(б) $k = 6$

(в) $k = 8$

Слика 4.12: Резултати за град Питсбург

4.6 Завршна разматрања

У овом поглављу развијена је похлепна хеуристика и метода променљивих околина за решавање k-јаке римске доминације, проблема код кога је и сама провера допустивости изазовна. Увођење концепта квазидопустивости омогућава ефикасно кретање простором претраге уз избегавање скупе експоненцијалне егзактне провере. Експерименти показују да VNS значајно побољшава иницијална похлепна решења, постиже оптималне или висококвалитетне резултате на инстанцама где су егзактне методе применљиве, и надмашује конкурентске приступе на великим графовима. Студија случаја са позиционирањем ватрогасних станица и возила у регионима, тако да читав граф буде безбедан у случају k истовремених пожара, илуструје директну применљивост предложене методе. Демонстрирано је да се и веома сложени доминацијски проблеми могу успешно решавати пажљиво дизајнираним метахеуристикама, као и примењивати на релевантне практичне проблеме.

Поглавље 5

Проблем минималне доминације на великим графовима

Проблем минималне доминације (енг. *Minimum Dominating Set*, MDS) представља основну, а самим тим и највише проучавану варијанту доминацијских проблема на графовима, како са теоријске, тако и са практичне стране. Као што је већ поменуто у уводном поглављу 1, важност овог проблема произлази из бројних примена у различитим областима, међу којима су и телекомуникације, друштвене мреже и биоинформатика. С обзиром на то да је проблем MDS НП-тежак, посебан изазов представља решавање овог проблема на великим графовима.

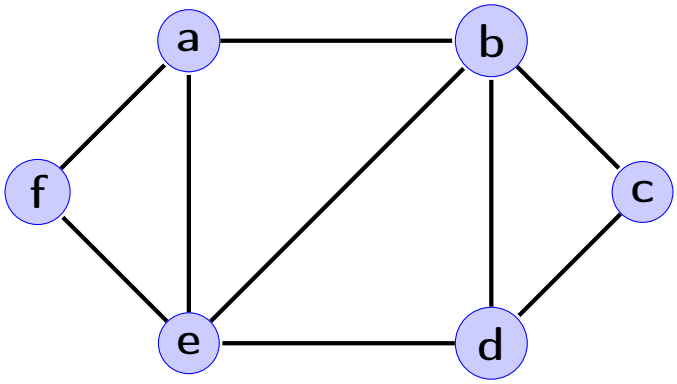
Иако се егзактне методе попут целобројног линеарног програмирања изузетно добро показују на мањим графовима, са порастом величине инстанце, долази до комбинаторне експлозије, што онемогућава њихову директну примену. Управо због тога, већина радова у литератури се бави развојем ефикасних хеуристичких и метахеуристичких приступа за овај проблем. У овом поглављу биће представљен нови хибридни алгоритам IRIS (енг. *Iterative Refinement via ILP Subproblems*), чија је основна идеја да комбинује снагу егзактних и ефикасност метахеуристичких метода. Заснован је на итеративном побољшању тренутног решења, кроз решавање пажљиво одабраних ILP потпроблема, који су довољно мали да их савремени решавачи могу обрадити, али и довољно информативни да обезбеде значајан напредак. Иако је у овом поглављу IRIS примењен на проблем MDS, предложени приступ има општу природу и може се прилагодити и за друге НП-тешке проблеме.

5.1 Дефиниција проблема

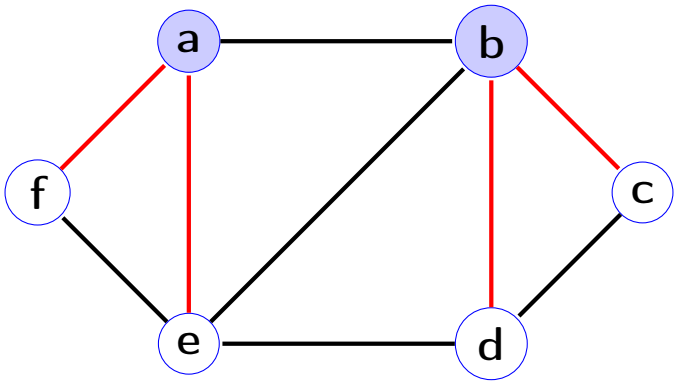
Проблем MDS се дефинише на неусмереном нетежинском графу $G = (V, E)$, где је V скуп чворова, а E скуп грана. Скуп $S \subseteq V$ назива се доминирајући ако за сваки чвор $v \in V \setminus S$ постоји бар један чвор $u \in S$ такав да је $\{u, v\} \in E$. Сам проналазак доминирајућег скупа није тежак проблем, јер скуп $S = V$ увек представља доминирајући скуп. Међутим, циљ проблема MDS је да се пронађе доминирајући скуп минималне кардиналности, што је, како је показано у одељку 1.1, НП-тежак проблем.

5.1.1 Пример инстанце

Пример инстанце проблема MDS приказан је на слици 5.1. Чворови су означени словима од a до f, док су гране приказане линијама. Постоји више оптималних решења за ову инстанцу, а једно од њих је приказано на слици 5.2. Чворови који припадају решењу су обојени плавом бојом, док су гране којима доминирају над преосталим чворовима обојене црвеном бојом.



Слика 5.1: Пример инстанце проблема MDS



Слика 5.2: Пример оптималног решења проблема MDS

5.2 Преглед литературе

Преглед литературе у овом поглављу обухвата радове посвећене развоју ефикасних алгоритама за решавање проблема MDS на великим графовима, као и радове који разматрају хибридне приступе за решавање НП-тешких проблема, са посебним осврtом на хибридизацију егзактних и метахеуристичких метода.

5.2.1 Проблем MDS на великим графовима

Доминација на графовима је формално дефинисана средином 20. века, али сам концепт доминације постојао је и бар један век раније. Године 1862. у књизи [64] разматран је проблем проналаска минималног броја краљица које могу да покрију шаховску таблу, тако да је свако поље или нападнуто или се на њему налази краљица. Овај проблем може се формулисати као проблем MDS на графу у коме су чворови поља шаховске табле, а гране повезују поља која краљица може да нападне. У књизи [14] из 1958. године, први пут је формално дефинисан појам доминације на графовима, мада не под тим називом. Са данашњим именом, проблем MDS се први пут појављује у књизи [97] из 1962. године. Веће интересовање за доминацијске проблеме јавља се од 1970-их година, поготово након прегледног рада [31], у коме су систематизовани резултати о доминацијским проблемима до тада. Од тог тренутка, доминацијски проблеми постају једни од најинтензивније проучаваних комбинаторних проблема на графовима, како са теоријске, тако и са практичне стране. Преглед познатих резултата може се наћи у књизи [59]. У овом одељку биће представљени само радови који се баве развојем ефикасних алгоритама за решавање проблема MDS на великим графовима.

Најбољи резултати на великим графовима до сада, постигнути су коришћењем

метахеуристичких приступа. Издвајају се радови [117] из 2024. и [99] из 2025. године. Оба рада су заснована на сличној идеји - користе врсту итеративне локалне претраге, у оквиру које постоје два различита оператора.

У раду [117], користе се оператори (2, 1)-замене и (3, 2)-замене, односно уклањају се 2, тј. 3 чвора из тренутног решења, а затим се додаје 1, тј. 2 чвора како би се поново добило допустиво решење. Иницијално решење се конструише уз помоћ похлепне хеуристике, а у току рада алгоритма користе се статистички подаци о фреквенцији појављивања чворова у решењима како би се усмеравала претрага.

У раду [99], оба оператора су (2, 1)-замене. Један је стандардан, а други врши процену вредности функције циља на основу локалне околине разматраног чвора. Дакле, не рачуна је егзактно, те на тај начин доприноси експлорацији читавог алгоритма. Такође користи податке о броју итерација у којима је чвор био део решења како би се усмерила претрага.

5.2.2 Хибридни приступи за решавање НП-тешких проблема

У последњих неколико деценија, развијени су бројни хибридни приступи који комбинују снагу егзактних метода са ефикасношћу метахеуристичких алгоритама. Преглед оваквих, али и хибридних приступа који комбинују метахеуристичке методе са другим хеуристичким техникама, може се наћи у раду [18].

Један од најпознатијих хибридних алгоритама је CMSA (енг. *Construct, Merge, Solve & Adapt*) [17]. Овај алгоритам функционише на принципу конструисања већег броја решења на пробабилистички начин, након чега се компоненте тих решења спајају у редуковану подинстанцу (потпроблем) која се решава егзактним ILP решавачем. У стандардној верзији, величина овог потпроблема се контролише механизмом „старења“ компоненти, што захтева пажљиво подешавање параметара. Као одговор на проблем осетљивости на параметре, развијен је *Adapt-CMSA* [5]. Супротно претходним верзијама, *Adapt-CMSA* уклања механизам старења (ресетовањем потпроблема у свакој итерацији) и уместо тога уводи механизам самоприлагођавања (енг. *self-adaptation*). Овај механизам динамички подешава пристрасност при конструкцији решења на основу перформанси решавача у претходним итерацијама, чиме се аутоматски одржава оптимална величина потпроблема без потребе за екстерним подешавањем.

Суштинска разлика између CMSA породице алгоритама и предложеног IRIS приступа је у начину дефинисања простора претраге. CMSA алгоритми имплицитно врше интензификацију око тренутног најбољег решења (пристрасном конструкцијом нових решења која су му мање или више слична у зависности од параметара), њихов потпроблем је увек резултат агрегације компоненти из популације пробабилистички генерисаних решења. Насупрот томе, IRIS не генерише скупове помоћних решења. Уместо тога, врши се декомпозиција тренутног решења кроз фиксирање одређених променљивих, док се одабир слободних променљивих врши на основу статистичке анализе структуре проблема, као и историје претраге.

Приступ који је структурно најближи IRIS алгоритму представљен је у раду [107]. Као и IRIS, овај рад предлаже општи оквир где се ILP потпроблеми користе за интензификацију претраге. Међутим, постоје две фундаменталне разлике које IRIS чине флексибилнијим и лакшим за примену.

Прва разлика се односи на стратегију дефинисања потпроблема. Алгоритам из рада [107] у сваком кораку врши партиционисање скупа свих променљивих на k дисјунктних подскупова, а затим их решава секвенцијално. Оваква „чврста“ подела има значајан недостатак: уколико су две променљиве зависне (нпр. морају истовремено променити вредност ради очувања допустивости или побољшања вредности функције циља), а додељене су различитим подскуповима, решавач неће моћи да пронађе

побољшање ни у једној итерацији јер је у сваком тренутку једна од њих фиксирана. Насупрот томе, IRIS не врши глобалну партицију, већ у свакој итерацији динамички конструише јединствен скуп слободних променљивих фокусирајући се на оне делове графа који су статистички идентификовани као нестабилни или обећавајући. Ово омогућава флексибилније претраживање околине без вештачких граница које намеће дисјунктна подела.

Друга кључна разлика је у примени машинског учења. Рад [107] се ослања на учење поткрепљивањем, које захтева велики скуп инстанци за тренирање и дуготрајну фазу обучавања модела пре самог решавања (енг. *offline learning*). С друге стране, иако је остављена могућност за коришћење оваквих приступа, IRIS превасходно користи учење о структури проблема током самог извршавања (енг. *online learning*). Коришћењем статистика (попут покривености чворова) које се ажурирају у реалном времену, IRIS се брзо адаптира на специфичности конкретне инстанце без потребе за претходним обучавањем или екстерним подацима.

5.3 Алгоритам IRIS

Као што је раније помињано, сви метахеуристички алгоритми се састоје од две главне компоненте: интензификације и диверзификације. Интензификација има за циљ да пронађе боља решења у околини тренутног решења, док диверзификација има за циљ да истражи нове области простора претраге. Уобичајено је процедура интензификације далеко скупља, односно захтева више временских ресурса. Често се реализује кроз локалну претрагу, која може бити ограничена на релативно малу околину тренутног решења, како би се обезбедила ефикасност. А чак и тада, у случају великих инстанци, може бити превише временски захтевна.

Савремени ILP решавачи, као што су `Gurobi` и `CPLEX`, користе бројне напредне технике које им омогућавају да ефикасно решавају, не само мале, већ и инстанце средње величине различитих НП-тешких проблема. Стога се природно намеће питање да ли је могуће искористити снагу ових решавача у оквиру метахеуристичког алгоритма. Управо на овој идеји је заснован нови хибридни алгоритам IRIS (енг. *Iterative Refinement via ILP Sub-problems*), који уместо локалне претраге, користи ILP потпроблеме за интензификацију претраге. У оквиру овог приступа, гради се и решава низ пажљиво одабраних ILP потпроблема, који су довољно мали да их савремени решавачи могу обрадити, али и довољно информативни да обезбеде значајан напредак. Потпроблеми се формирају на основу тренутног решења, тако што се одређени број променљивих фиксира на вредности из тог решења, док се остале остављају слободним. Решавањем тако конструисаних ILP модела, добијају се локална побољшања тренутног решења. Процес се понавља све док се не испуни неки од критеријума заустављања. Псеудокод алгоритма IRIS дат је у алгоритму 11, а у наставку ће бити детаљније објашњене његове појединачне компоненте.

Алгоритам 11 IRIS

Улаз: инстанца проблема I, ILP модел M, временско ограничење потпроблема t_{sub}, број слободних променљивих у потпроблему n_{free}
Излаз: решење s_{best}

```
1: s_best ← initialize(I)
2: while ! terminationCriteriaSatisfied() do
3:     F ← selectFixedVars(s_best, I, n_free)
4:     M_sub ← buildSubmodel(M, F, s_best)
5:     s' ← solveILP(M_sub, t_sub)
6:     if s' improves s_best according to acceptance criteria then
7:         s_best ← s'
8:     end if
9: end while
```

5.3.1 Иницијализација

Почетно решење може бити конструисано на различите начине: похлепном хеуристиком, неком од метахеуристичких метода, случајним или тривијалним решењем. Алгоритам IRIS функционише без обзира на избор стратегије иницијализације. Ипак, као и код осталих оптимизационих алгоритама, квалитет почетног решења може значајно утицати на брзину конвергенције и квалитет крајњег решења. Добра иницијализација често омогућава ефикасније усмеравање претраге и експлоатацију обећавајућих области простора претраге у раним фазама рада алгоритма.

5.3.2 Стратегија фиксирања променљивих

У свакој итерацији алгоритма, бира се скуп променљивих које ће бити фиксиране на вредности из тренутног решења. Остале променљиве остају слободне и могу мењати своје вредности током решавања ILP потпроблема. Величина скупа фиксираних, односно слободних променљивих, представља важан параметар алгоритма. Мањи потпроблеми се брже решавају, али могу бити мање информативни, док већи потпроблеми могу обезбедити значајнија побољшања, али њихово решавање може захтевати више времена. Поред тога, сам састав скупа фиксираних променљивих F има значајан утицај на квалитет и време решавања ILP потпроблема. Политика избора скупа F може бити:

- насумична – на случајан начин се бирају променљиве које ће бити фиксиране;
- хеуристичка – променљиве се оцењују на основу релевантних карактеристика, као што су степен чвора, степен централности, и слично, а затим се бирају најбоље оцењене променљиве;
- статистичка – током рада алгоритма прикупљају се статистички подаци о понашању променљивих, као што су фреквенција појављивања у решењима, и на основу тих података се бирају променљиве које ће бити фиксиране;
- вођена машинским учењем – обучава се модел машинског учења, у току рада самог алгоритма (енг. *online learning*) или пре тога (енг. *offline learning*), који предвиђа које променљиве треба фиксирати на основу карактеристика инстанце, тренутног решења и стања претраге.

Наравно, могуће је и комбиновати више стратегија.

5.3.3 Формирање и решавање потпроблема

ILP потпроблем се формира на основу оригиналног ILP модела M и скупа фиксираних променљивих F. За сваку променљиву $x_i \in F$, додаје се ограничење $x_i = s_{best}[i]$, где је $s_{best}[i]$ вредност променљиве x_i у тренутном решењу s_{best}. Остале променљиве, односно оне ван скупа F, остају слободне.

Важно је истаћи разлику у начину дефинисања простора претраге потпроблема између предложеног IRIS алгоритма и сродног CMSA приступа. Иако се у оба случаја потпроблем своди на решавање ILP модела са додатним ограничењима која фиксирају одређене променљиве, механизам избора слободних променљивих је другачији. Код CMSA алгоритма, простор претраге се формира *конструктивно*, агрегацијом компоненти из скупа пробабилистички генерисаних решења (тзв. *Merge* корак). Иако се код комбинаторних проблема овај оператор често реализује као унија скупова компоненти, он је концептуално шири и зависи од природе проблема. Променљива x_i постаје слободна у ILP потпроблему само ако скуп спојених компоненти садржи обе њене вредности ($x_i = 0$ и $x_i = 1$). У супротном, она се имплицитно фиксира додавањем једнакосног ограничења на једину присутну вредност. Насупрот томе, IRIS не захтева популацију решења нити операцију агрегације. Уместо тога, алгоритам полази од једног решења и експлицитно дефинише скуп фиксираних променљивих на основу стратегије описане у претходној секцији, док преостале променљиве остају слободне. Овакав приступ омогућава прецизнију контролу над величином и структуром потпроблема без зависности од стохастичке разноликости генерисаних решења.

Укупан број ограничења у потпроблему се може смањити, уколико нека од ограничења из оригиналног модела постану тривијално задовољена након фиксирања променљивих. Тако конструисан ILP потпроблем се затим решава уз помоћ ILP решавача, са временским ограничењем t_{sub}. Као и обично, решавач враћа најбоље пронађено решење у датом временском року. Није неопходан проналазак оптималног решења потпроблема, јер и делимично побољшање тренутног решења може бити корисно.

5.3.4 Критеријуми прихватања

Након решавања ILP потпроблема, добија се ново решење s'. Као критеријум прихватања новог решења могу се користити различити приступи, устаљени у метахеуристичким алгоритмима:

- само побољшавајућа решења – ново решење се прихвата само ако је боље од тренутног решења, у смислу вредности функције циља, или по неком секундарном критеријуму квалитета;
- прихватање са вероватноћом – ново решење се прихвата са одређеном вероватноћом, која може зависити од разлике у квалитету између новог и тренутног решења, као и од тренутне фазе рада алгоритма;
- адаптивно – критеријум прихватања се прилагођава током рада алгоритма, на основу динамичке анализе напретка претраге.

5.3.5 Критеријуми заустављања

Услов заустављања алгоритма може бити дефинисан на произвољан начин. Неки од најчешће коришћених критеријума су:

- максимално дозвољено време извршавања;

- максималан број итерација;
- максималан број итерација без побољшања решења;
- достигнуто довољно добро решење, у смислу вредности функције циља.

5.4 Алгоритам IRIS за проблем MDS

У претходној секцији представљен је општи оквир хибридног алгоритма IRIS који се може применити на широк спектар НП-тешких проблема за које постоје ILP формулације. У овом одељку биће детаљније објашњена примена овог алгоритма на проблем MDS. Дакле, биће описана специјализација појединачних компоненти алгоритма за овај проблем.

5.4.1 Иницијализација

Почетно решење је конструисано похлепним алгоритмом заснованим на максимизацији броја недоминираних суседа. Овај алгоритам по својој природи увек производи допустиво решење. На тај начин, почетна тачка претраге се налази у допустивом простору, што омогућава да све касније итерације задрже ту особину.

Пре покретања алгоритма, могуће је извршити различите технике претпроцесирања које смањују димензионалност проблема.

Претпроцесирање

Извршени су експерименти са две различите врсте претпроцесирања. Прва врста је једноставна и коришћена је и у раду [117]. Састоји се од три правила:

- ако постоји изоловани чвор, он мора бити део доминирајућег скупа, па се може одмах додати у решење;
- ако постоји чвор са степеном 1, он мора бити доминиран од стране свог јединог суседа, па се он може избацити, а тај сусед додати у решење;
- ако постоји троугао у графу, односно три чвора међусобно повезана, од којих су два степена 2, та два чвора се могу избацити из графа, а трећи чвор се може додати у решење.

Друга врста претпроцесирања је сложенија и није познато да ли је раније коришћена у литератури. Заснива се на следећој идеји везаној за затворене околине чворова.

Теорема 1. Ако је $N[u] \subset N[v]$, онда за сваки доминацијски скуп S где је $u \in S$, постоји доминацијски скуп S' такав да $u \notin S'$, $v \in S'$, $|S'| \leq |S|$.

Доказ. Како је $N[u] \subset N[v]$, сваки чвор који u доминира, доминира и v. Стога, замена u са v у доминацијском скупу S чува доминацију свих чворова које је u доминирао. Штавише, v може да доминира и додатне чворове из $N[v] \setminus N[u]$.

Нека је S' скуп добијен заменом u са v, односно $S' = (S \setminus \{u\}) \cup \{v\}$. Тада је $|S'| = |S|$ и S' је доминирајући скуп. Након ове замене неки чворови из S' могу постати сувишни, па се могу избацити из скупа. Ако се они уклоне, добија се доминацијски скуп S', такав да је $|S'| \leq |S|$. □

Последица 1. Ако је $N[u] \subset N[v]$, онда чвор u не мора бити део минималног доминацијског скупа.

Иако је ова врста претпроцесирања концептуално софистицираније, у експерименталним резултатима (видети секцију 5.5) није дала боље резултате. Ово може бити последица глобалнијег карактера овог правила, који не одговара добро локалној природи похлепног алгоритма.

Похлепни алгоритам

Након претпроцесирања, иницијално решење се конструише похлепним алгоритмом. У свакој итерацији, бира се чвор који покрива највећи број недоминираних суседа. Псеудокод дат је у алгоритму 12.

Алгоритам 12 Похлепни алгоритам

Улаз: граф G, скуп забрањених чворова F, скуп обавезних чворова M
Излаз: доминацијски скуп S

```
1: S ← M
2: U ← V \ N[S]
3: while U ≠ ∅ do
4:     for v ∈ V \ (S ∪ F) do
5:         score[v] ← countUndominatedNeighbors(v, G, U)
6:     end for
7:     v* ← argmax_v(score[v])
8:     S ← S ∪ {v*}
9:     U ← U \ N[v*]
10: end while
11: return S
```

Након сваке итерације ажурира се број недоминираних суседа само за чворове у околини изабраног чвора. Коришћена је структура података макс-хип (енг. *max-heap*) која омогућава ефикасан избор наредног чвора, у времену $O(1)$. Након избора чвора, ажурирају се вредности суседа и њихових суседа, што захтева $O(d^2)$ времена, где је d просечан степен чвора у графу. Пошто свако ажурирање захтева $O(\log n)$ времена, укупна временска сложеност једне итерације је $O(d^2 \log |V|)$. Како се у свакој итерацији избацује читава затворена околина изабраног чвора, укупан број итерација је $O(|V|/d)$. Дакле, укупна временска сложеност похлепног алгоритма је $O(|V|d \log |V|)$, што је нарочито ефикасно за ретке графове, где је d знатно мање од $|V|$.

Постпроцесирање

Након конструисања почетног решења, могуће је извршити постпроцесирање како би се додатно смањила величина доминацијског скупа. С обзиром на то да је решење вероватно субоптимално, могуће је да неки чворови у доминацијском скупу нису неопходни. У оквиру постпроцесирања избацују се сви чворови за које важи да сви чворови из њихове затворене околине имају бар два суседа у тренутном решењу. Другим речима, ако се такав чвор избаци из доминацијског скупа, сви његови суседи, а и он сам, и даље имају бар једног суседа у доминацијском скупу, односно није нарушен услов доминације.

5.4.2 Стратегија фиксирања променљивих

Као што је већ поменуто, у свакој итерацији алгоритма бира се скуп променљивих које ће бити фиксиране на вредности из тренутног решења, док се преостале променљиве остављају слободним. У овом случају, променљиве одговарају чворовима графа, а

вредност променљиве је 1 ако чвор припада доминацијском скупу, односно 0 у супротном. Чворови који ће бити фиксирани бирају се на основу мере покривености, која изражава колико пута је дати чвор доминиран од стране чворова из тренутног решења. Интуитивно, чворови чије затворене околине имају већу покривеност, мање су критични за очување доминације, па је мања вероватноћа да ће остати фиксирани. Коришћена је стратегија налик рулетској селекцији, где је вероватноћа избора чвора обрнуто пропорционална његовој покривености.

Поред тога, финална вероватноћа је добијена комбиновањем са историјским вероватноћама које прате стабилност чворова током итерација алгоритма. Након сваког покушаја побољшања решења, ове, историјске, вероватноће ажурирају се у зависности од исхода. Уколико задржавање неких фиксираних чворова у решењу није довело до побољшања, њихова вероватноћа задржавања се благо смањује. Са друге стране, благо се повећава вероватноћа фиксирања преосталих чворова који нису били одабрани. Овај механизам ажурирања вероватноћа концептуално подсећа на механизам испаравања феромона код оптимизације колонијом мрава (енг. *Ant Colony Optimization*) или на механизам „старења“ компонената код CMSA приступа. Међутим, кључна разлика је у томе што се овде смањивање вероватноће користи за дефинисање чврстих ограничења (фиксирање променљивих) у математичком моделу, а не директно за конструкцију решења. На тај начин се постепено формира стабилна статистика о томе који чворови доприносе квалитету решења, што води ка конзистентнијем избору променљивих у наредним итерацијама.

Битан параметар алгоритма је број слободних променљивих n_{free}, односно величина ILP потпроблема. Број слободних променљивих зависи од величине графа. За мање графове, из тренутног решења се фиксира једна трећина променљивих, док преостале остају слободне. За веће графове се поставља граница на максималан број слободних променљивих на 30 хиљада. Ова вредност је одређена емпиријски и представља компромис између величине простора претраге и ефикасности ILP решавача. Иако већи број слободних променљивих теоријски нуди већу шансу за излазак из локалног оптимума, експерименти су показали да изнад ове границе, за задато временско ограничење t_{sub}, долази до комбинаторне експлозије коју комерцијални решавачи не могу ефикасно да савладају, што резултира стагнацијом претраге.

Треба напоменути да постоје приступи, попут *Adapt-CMSA* [5], који користе адаптивни механизам за динамичко одређивање величине потпроблема на основу перформанси решавача у претходним итерацијама. Иако би се слична логика могла применити и на параметар n_{free} (нпр. повећање броја слободних променљивих након успешног побољшања решења), у овој верзији IRIS алгоритма одлучено је да се користи фиксна горња граница како би се смањила комплексност алгоритма и број контролних параметара. Увођење адаптивног механизма за n_{free} представља правац за будућа истраживања и даља побољшања перформанси алгоритма.

5.4.3 Формирање и решавање потпроблема

Потпроблем који се решава у свакој итерацији формулисан је као проблем целобројног линеарног програмирања. За сваки чвор $v_i \in V$ дефинише се бинарна променљива x_i, која је 1 ако чвор v_i припада доминацијском скупу, односно 0 у супротном. Циљ је минимизовати број изабраних чворова, уз услов да сваки чвор буде доминиран. ILP модел је следећи:

$$\min \sum_{v_i \in V} x_i \tag{5.1}$$

уз услове

$$x_i + \sum_{v_j \in N(v_i)} x_j \geq 1, \quad \forall v_i \in V \tag{5.2}$$

$$x_i \in \{0, 1\}, \quad \forall v_i \in V \tag{5.3}$$

Овај модел се модификује у свакој итерацији алгоритма IRIS, тако што се за сваки чвор v_i који припада скупу фиксираних чворова F, додаје ограничење $x_i = s_{best}[i]$, где је $s_{best}[i]$ вредност променљиве x_i у тренутном решењу s_{best}.

У складу са општим принципима редукције модела наведеним у одељку 5.3.3, фиксирање променљивих значајно поједностављује простор претраге. Када је $s_{best}[i] = 1$, ограничење (5.2) за чвор v_i, као и за све његове суседе, постаје тривијално задовољено. Међутим, експлицитно детектовање и уклањање свих таквих ограничења пре позива решавача захтевало би додатне итерације кроз структуру графа, што би успорило алгоритам. Уместо тога, ослањамо се на *presolve* фазу `CPLEX` решавача, која је високо оптимизована за овакве операције и која интерно елиминише редундантна ограничења много ефикасније него што би се то постигло екстерним претпроцесирањем.

Поред тога, у модел се додаје ограничење везано за вредност функције циља, које осигурава да се не разматрају решења лошија од тренутно најбољег:

$$\sum_{v_i \in V} x_i \leq |s_{best}| \tag{5.4}$$

Као почетно решење (енг. *warm start*) ILP решавачу се прослеђује тренутно најбоље решење s_{best}, што убрзава конвергенцију и често води до бољег локалног побољшања.

ILP модел се у свакој итерацији решава помоћу `CPLEX` решавача верзије 22.1, уз временско ограничење од 60 секунди. Ова вредност је одабрана као стратешки компромис између квалитета оптимизације појединачног потпроблема и укупног броја итерација које алгоритам може да изврши у оквиру глобалног временског ограничења од 1000 секунди. Повећање времена за решавање потпроблема би, иако потенцијално корисно за локално побољшање, директно смањило укупан број итерација, а тиме и могућност алгоритма да истражи различите делове простора претраге кроз промену скупа фиксираних променљивих. Такође, треба нагласити да је величина ILP потпроблема експлицитно ограничена параметром n_{free} (на максимално 30 хиљада), независно од укупне величине графа. Због тога комплексност решавања потпроблема не расте линеарно са величином инстанце, што чини фиксно временско ограничење оправданим избором који не захтева скалирање у односу на број чворова у графу.

Наравно, ради максимизације перформанси алгоритма IRIS, у будућим истраживањима биће спроведено ригорозно подешавање параметара t_{sub} и n_{free}, као и испитивање адаптивних стратегија за њихово динамичко прилагођавање током рада алгоритма.

5.4.4 Критеријуми прихватања

У овом случају, ново решење се прихвата само ако је боље од тренутног решења, односно ако је добијени скуп мање кардиналности. У ILP модел је додато ограничење које осигурава да се не разматрају решења лошија од тренутно најбољег. Дакле, ако ILP решавач пронађе ново допустиво решење, оно је или исте величине као и тренутно најбоље, или боље. Прихватају се само боља решења. С обзиром на то да је почетно решење допустиво, те да је ILP модел формиран тако да задржава све доминацијске услове, свако ново прихваћено решење је такође допустиво.

5.4.5 Критеријуми заустављања

Алгоритам се зауставља када је испуњен бар један од следећа два услова:

- максимално дозвољено време извршавања од 1000 секунди – овај критеријум је стандардно коришћен у литератури [117, 99];
- максималан број итерација од 100 – овај критеријум је уведен ради скраћивања укупног времена извршавања на великом броју инстанци.

5.5 Експериментални резултати

Експерименти су изведени са циљем процене исправности и ефикасности предложеног IRIS алгоритма. Резултати су упоређени са два најбоља метахеуристичка приступа за проблем MDS на великим графовима, описана у секцији 5.2. Алгоритам је имплементиран у програмском језику Python. Сви експерименти су извршени у режиму са једном нити, на рачунару са Intel Core i9-11900 процесором са радним тактом од 2.5GHz, уз ограничење од 16GB RAM меморије по извршавању, под оперативним системом Ubuntu 24.04.

5.5.1 Инстанце проблема

За разлику од радова [117] и [99], који су фокусирани на постизање најбољих резултата (енг. *state-of-the-art*) и тестирани су на стотинама инстанци, овде је циљ био показивање опште применљивости алгоритма IRIS и његовог потенцијала за рад на великим инстанцама. Стога је одабран подскуп инстанци коришћених у литератури који се састоји из целокупног SNAP [78] скупа и дела скупа DIMACS10 [8].

- SNAP скуп садржи реалне, велике графове из различитих домена, као што су друштвене мреже, мреже сарадње и веб графови. Коришћене су све SNAP инстанце као у радовима [117] и [99].
- DIMACS10 скуп садржи реалне и синтетичке графове који су коришћени у 10. DIMACS такмичењу за партиционисање и кластеровање графова. Из овог скупа је одабран подскуп од 14 инстанци.

Још једна разлика у односу на наведене радове је број покретања алгоритма по инстанци. Уместо 10 покретања као у тим радовима, овде је извршено по једно покретање по инстанци како би се смањило укупно време извршавања.

5.5.2 Анализа стабилности алгоритма

Како би се установило да је методологија једног покретања по инстанци оправдана, спроведена је додатна анализа како би се испитала стабилност предложеног IRIS алгоритма и валидирао овакав приступ. За ову сврху одабран је репрезентативан подскуп од 10 инстанци које покривају различите величине (од 10 хиљада до 3.7 милиона чворова) и различите топологије (друштвене мреже, веб графови, биолошке мреже, путна инфраструктура и вештачки графови).

На свакој од ових инстанци алгоритам је покренут 10 пута независно. У табели 5.1 приказани су сумарни резултати ове анализе, укључујући просечну вредност (μ), најбољу пронађену вредност, стандардну девијацију (σ), као и коефицијент варијације (CV), дефинисан као однос стандардне девијације и средње вредности ($\frac{\sigma}{\mu} \times 100\%$).

Табела 5.1: Анализа стабилности алгоритма IRIS

Инстанца	Просек (μ)	Мин.	Стд. дев. (σ)	CV (%)
Soc-Slashdot0902	15305.0	15305	0.0	0.000
belgium.osm	485909.5	485843	67.3	0.014
caidaRouterLevel	40522.0	40522	0.0	0.000
cnr-2000	22006.0	22006	0.7	0.003
ecology1	250075.5	250030	59.6	0.024
cit-Patents	634982.0	634952	58.2	0.009
rgg-n-2-20-s0	89414.5	89385	41.0	0.046
web-Google	79698.0	79698	0.0	0.000
Amazon0601	42363.5	42311	16.1	0.038
p2p-Gnutella31	12582.0	12582	0.0	0.000
Просек				**0.013**

Резултати показују изузетну стабилност IRIS алгоритма. На 4 од 10 инстанци, укључујући и велику инстанцу `web-Google`, стандардна девијација резултата је 0, што значи да алгоритам у свих 10 покретања проналази решење исте кардиналности. Максимални забележени коефицијент варијације износи свега 0.046% (на инстанци `rgg-n-2-20-s0`), док је просечни коефицијент варијације на целом узорку 0.013%.

Овако низак ниво варијације потврђује да стохастички елементи алгоритма имају занемарљив утицај на коначан квалитет решења. Сходно томе, једноструко покретање алгоритма пружа поуздану процену његових перформанси, чиме је оправдан приступ коришћен у главном делу експеримената.

5.5.3 Резултати

У табелама 5.2 и 5.3 су приказани резултати експеримената на скуповима инстанци описаним у претходној секцији. Структура табела је следећа. Свака табела се састоји од шест колона. У прве три колоне се налазе подаци о инстанци: назив, број чворова $|V|$ и број грана $|E|$. У наредне две колоне су резултати алгоритма DemDS [99] – минимална и просечна вредност кардиналности доминацијског скупа из 10 покретања. Исто тако, у следеће две колоне су резултати алгоритма DmDS [117]. У последњој колони се налазе резултати предложеног IRIS алгоритма. Најбољи резултати у сваком реду су подебљани.

Резултати на SNAP скупу

Експериментални резултати на SNAP скупу приказани су у табели 5.2, чија је структура објашњена на почетку ове секције.

На основу резултата приказаних у табели 5.2, може се закључити да предложени IRIS алгоритам показује конкурентне перформансе у односу на најбоље постојеће метахеуристичке приступе за проблем MDS. У наставку су издвојени неки од најважнијих закључака:

- Од укупно 22 инстанце у SNAP скупу, IRIS постиже најбоље резултате на 13 инстанци, од чега су на 11 инстанци резултати једнаки најбољим постојећим, а на 2 инстанце су бољи. На преосталих 9 инстанци IRIS постиже резултате који су близу најбољих. Конкретно, просечна разлика у односу на најбољи резултат износи 0.17%, док је максимална разлика 2.18%.

- DemDS постиже најбоље резултате на 12 инстанци, од чега је на 2 инстанце строго

Табела 5.2: Нумерички резултати на скупу SNAP

инстанца			DemDS		DmDS		IRIS
назив	|V|	|E|	минимум	просек	минимум	просек	
Amazon0302	262111	899792	**35599**	35605.6	35600	35601.9	35691
Amazon0312	400727	2349869	45485	45490.6	**45475**	45480.7	45601
Amazon0505	410236	2439437	47308	47315.4	**47303**	47309.1	47405
Amazon0601	403394	2443408	42285	42290.6	**42279**	42282.1	42356
Cit-HepPh	34546	420877	**3078**	3078.6	**3078**	3078.9	3080
Cit-HepTh	27770	352285	**2935**	2935.9	2936	2936.1	2936
cit-Patents	3774768	16518947	621620	621661.0	**621518**	621534.7	635079
Email-EuAll	265214	364481	**18181**	18181.0	**18181**	18181.0	**18181**
p2p-Gnutella04	10876	39994	**2227**	2227.0	**2227**	2227.0	**2227**
p2p-Gnutella24	26518	65369	**5418**	5418.0	**5418**	5418.0	**5418**
p2p-Gnutella25	22687	54705	**4519**	4519.0	**4519**	4519.0	**4519**
p2p-Gnutella30	36682	88328	**7169**	7169.0	**7169**	7169.0	**7169**
p2p-Gnutella31	62586	147892	**12582**	12582.0	**12582**	12582.0	**12582**
Soc-Epinions1	75879	405740			**15734**	15734.0	**15734**
Soc-Slashdot0811	77360	469180			**14312**	14312.0	**14312**
Soc-Slashdot0902	82168	504230			**15305**	15305.0	**15305**
web-BerkStan	685230	6649470	28445	28453.2	**28438**	28438.8	28479
web-Google	875713	4322051	79699	79699.0	79699	79699.0	**79698**
web-NotreDame	325729	1090108	23734	23734.0	23734	23734.0	**23732**
web-Stanford	281903	1992636	**13197**	13198.8	**13197**	13198.8	13237
Wiki-Talk	2394385	4659565	**36960**	36960.0	**36960**	36960.0	**36960**
Wiki-Vote	7115	100762	**1116**	1116.0	**1116**	1116.0	**1116**

најбољи. На преосталим инстанцама, изузимајући 3 за које резултати нису доступни, постиже резултате веома близу најбољих, са максималним одступањем од 0.02%.

- DmDS постиже најбоље резултате на 18 инстанци, од чега је на 5 инстанци строго најбољи. На преосталим инстанцама постиже резултате веома близу најбољих, са максималним одступањем од 0.03%.

Сумарно, сва три упоређена алгоритма показују сличне перформансе на SNAP скупу, са малим релативним разликама у квалитету добијених решења. Највеће одступање од најбољег резултата за алгоритам IRIS се догађа на инстанци `cit-Patents`, која је и највећа инстанца у овом скупу, са преко 3.7 милиона чворова и 16.5 милиона грана. Ово указује на то да постоји простор за даља побољшања у раду алгоритма на изузетно великим графовима.

Резултати на DIMACS10 скупу

Резултати експеримената на DIMACS10 скупу приказани су у табели 5.3, са структуром објашњеном на почетку ове секције.

Из добијених нумеричких резултата приказаних у табели 5.3, могу се извући следећи закључци:

- Алгоритам IRIS постиже најбоље резултате на 8 од укупно 14 инстанци. Међу њима, на 6 инстанци резултати су једнаки најбољим постојећим, док су на 2 инстанце бољи. На преосталим инстанцама, резултати су релативно близу најбољих, са просечном разликом од 2.68% и максималном разликом од 24.55%. Највеће одступање се јавља

Табела 5.3: Нумерички резултати на скупу DIMACS10

инстанца			DemDS		DmDS		IRIS
назив	|V|	|E|	минимум	просек	минимум	просек	
as-22july06	22963	48436	**2026**	2026.0	**2026**	2026.0	**2026**
belgium.osm	1441295	1549970			**468909**	468925.8	485748
caida*Level	192244	609066	40523	40523.0	40523	40523.0	**40522**
citat*eseer	268495	1156647	**43412**	43412.0	**43412**	43412.0	**43412**
cnr-2000	325557	2738969			22010	22010.4	**22006**
coAut*seer	227320	814134	**33197**	33197.0	**33197**	33197.0	**33197**
coAut*DBLP	299067	977676	**43978**	43978.0	**43978**	43978.0	**43978**
cond-*2005	39577	175691	**5664**	5664.0	**5664**	5664.0	**5664**
ecology1	1000000	1998000	**200883**	201197.3	201245	201609.2	250194
kron-*logn16	55321	2456071	**3885**	3885.0	**3885**	3885.0	**3885**
luxembourg.osm	114599	119666	**37751**	37753.9	**37751**	37752.2	37876
rgg-n-2-17-s0	131070	728753	**12310**	12314.9	12319	12321.8	12482
rgg-n-2-19-s0	524284	3269766	**44347**	44361.0	**44347**	44362.5	45180
rgg-n-2-20-s0	1048575	6891620	84592	84609.3	**84544**	84567.0	89466

на инстанци `ecology1`, која је једна од већих у овом скупу, са милион чворова и скоро два милиона грана. Сва остала одступања су значајно мања.

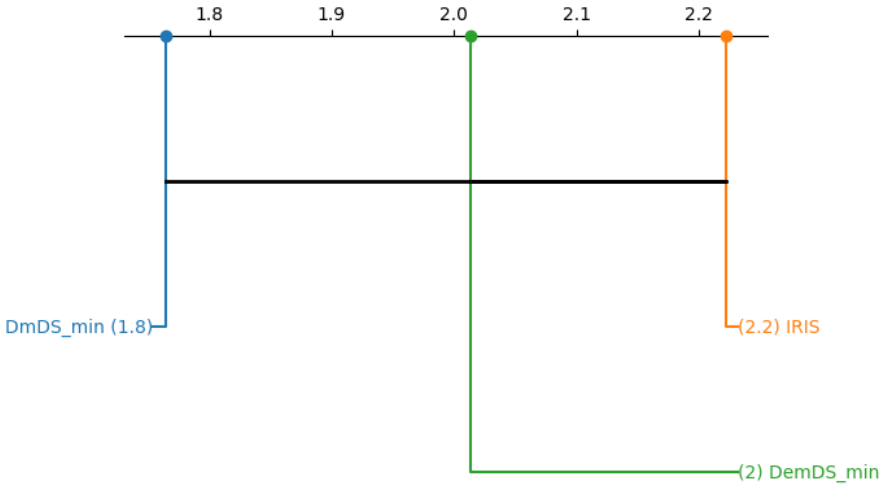
- Алгоритам DemDS постиже најбоље резултате на 10 инстанци, од чега су на 2 инстанце строго најбољи. На преосталим инстанцама (не рачунајући две за које резултати нису доступни), постиже резултате близу најбољих, са максималним одступањем од 0.06%.
- Алгоритам DmDS такође постиже најбоље резултате на 10 инстанци, од чега је на 2 инстанце строго најбољи. И овај приступ постиже резултате близу најбољих на преосталим инстанцама, са максималним одступањем од 0.18%.

Укупно, поново се може закључити да три упоређена приступа показују сличне перформансе на DIMACS10 скупу, са малим релативним разликама у квалитету добијених решења. Изузетак је инстанца `ecology1`, на којој IRIS показује лошије резултате у односу на најбоље постојеће приступе. У овом случају, с обзиром на то да на неколико других, чак и већих инстанци, IRIS постиже резултате који су ближи најбољим, може се претпоставити да је специфична структура графа на овој инстанци изазвала потешкоће за предложени приступ. Наиме, за разлику од друштвених и веб мрежа које су густе и имају својство „малог света“ (енг. *small-world graphs*), `ecology1` је инстанца која има значајно другачију топологију, а то је дводимензиони граф решетке (енг. *2D grid graph*). Таква структура негативно утиче, како на похлепни алгоритам који се користи за конструкцију почетног решења, тако и на стратегију избора фиксираних чворова, пошто су оба примарно заснована на мери покривености чворова.

5.5.4 Статистичка анализа

Извршена је статистичка анализа добијених резултата коришћењем Фридмановог теста [47]. Анализирани су резултати сва три алгоритма: DemDS, DmDS и IRIS. Нулта хипотеза претпоставља статистички једнаке перформансе свих алгоритама. На основу резултата теста, ни на SNAP ($p = 0.0755$), ни на DIMACS10 ($p = 0.3433$) скупу, није могуће одбацити нулту хипотезу са нивоом значајности 0.05. Стога, може се закључити да нема статистички значајне разлике у перформансама трију анализираних алгоритама.

Поред тога, извршена је и статистичка анализа на свим инстанцама заједно. У том случају, нулта хипотеза је одбачена ($p = 0.0288$). Након тога, примењен је Неменјијев пост-хок тест [93] како би се утврдило који се парови алгоритама статистички разликују. Добијени резултати су приказани на слици 5.3 у облику графика критичне разлике (енг. *critical difference*, CD). Како је и раније помињано, на овим графицима се алгоритми позиционирају дуж хоризонталне осе у складу са својим просечним рангом, док критична разлика представља минималну разлику у просечном рангу која је потребна да би се они сматрали статистички различитим.



Слика 5.3: График критичне разлике за све инстанце из скупова SNAP и DIMACS10

На основу резултата приказаних на слици 5.3, може се приметити да алгоритам DmDS има најбољи просечни ранг, а потом следе DemDS и IRIS. Међутим, иако постоји разлика у просечном рангу алгоритама, та разлика није довољно велика да би се сматрала статистички значајном. Треба нагласити и да су поређене минималне, односно најбоље вредности које су два конкурентска приступа постигла из 10 покретања, што је и назначено на слици. Са друге стране, IRIS алгоритам је покренут само једном по инстанци, те је коришћен његов једини резултат. Поред тога, за инстанце за које резултати нису доступни за DemDS алгоритам, недостајуће вредности су замењене великим вредностима (нпр. милион) како би се омогућило рангирање у оквиру статистичке анализе.

5.5.5 Анализа утицаја појединачних компоненти алгоритма

Спроведена је анализа утицаја појединачних компоненти алгоритма IRIS на квалитет добијених решења. Циљ ове анализе је изолација доприноса појединачних компоненти алгоритма како би се потврдила оправданост њиховог укључивања у коначни дизајн. Испитано је следеће:

- Утицај почетног решења добијеног похлепним алгоритмом у односу на тривијално почетно решење (сви чворови укључени у доминацијски скуп).
- Утицај ILP решавача у односу на случај када се потпроблеми решавају помоћу рандомизоване верзије похлепног алгоритма који се користи за конструкцију почетног решења.
- Утицај претпроцесирања и постпроцесирања.

Експерименти су изведени на истом репрезентативном подскупу од 10 инстанци који је коришћен у анализи стабилности алгоритма (одељак 5.5.2). Резултати ове анализе су приказани у табели 5.4. Алгоритам је извршен по једном на свакој инстанци у четири различите конфигурације:

- **Init: trivial** – почетно решење је тривијално (сви чворови укључени у доминацијски скуп);
- **Sub: rand-greedy** – потпроблеми се решавају помоћу рандомизоване верзије похлепног алгоритма (са вероватноћом 0.5 се бира најбољи следећи чвор, а у осталим случајевима се бира насумично између 10 најбољих);
- **No pre/post** – без претпроцесирања и постпроцесирања;
- **Full** – пуна верзија алгоритма IRIS.

Табела 5.4: Анализа утицаја појединачних компоненти алгоритма IRIS

Инстанца	Init: trivial	Sub: rand-greedy	No pre/post	Full
p2p-Gnutella31	**12582**	**12582**	**12582**	**12582**
soc-Slashdot0902	**15305**	15306	**15305**	**15305**
belgium.osm	486428	487377	487924	**485748**
caidaRouterLevel	**40522**	40852	40523	**40522**
cnr-2000	**22006**	22073	**22006**	**22006**
ecology1	**244177**	250334	249087	250194
cit-Patents	653592	640107	661215	**635079**
rgg-n-2-20-s0	88464	100954	**87940**	89466
web-Google	**79698**	79953	79703	**79698**
amazon0601	42339	44172	**42329**	42356
Просек	168511.3	169371	169861.4	**167295.6**

Из резултата приказаних у табели 5.4, могу се извући следећи закључци:

- Пуна верзија алгоритма IRIS постиже најбоље резултате на 7 од 10 инстанци. Поред тога, постиже најбољу просечну вредност на целом скупу инстанци. Важно је истаћи да само ова конфигурација постиже најбољи резултат на инстанци `cit-Patents`, која је највећа из оба коришћена скупа (SNAP и DIMACS10).
- Алгоритам са тривијалним почетним решењем постиже најбоље резултате на 6 инстанци. На 5 од тих 6, резултати су једнаки онима из пуне верзије алгоритма. Једина инстанца на којој ова конфигурација постиже бољи резултат од пуне верзије је `ecology1`, што потврђује претходно наведену тврдњу да похлепни алгоритам за конструкцију почетног решења вероватно није најпогоднији за ову специфичну топологију графа.
- Коришћење рандомизоване верзије похлепног алгоритма за решавање потпроблема доводи до најбољег резултата само на инстанци `p2p-Gnutella31`, где су сви резултати једнаки. Дакле, коришћење ILP решавача за потпроблеме је од изузетне важности за квалитет коначног решења.
- Конфигурација без претпроцесирања и постпроцесирања постиже најбоље резултате на 5 инстанци. Очекивано, ова верзија алгоритма је спорија од осталих због већег броја чворова који се морају обрадити.

5.6 Завршна разматрања

У оквиру овог поглавља развијен је хибридни алгоритам IRIS који комбинује решавање ILP модела са идејама метахеуристичке претраге кроз итеративну конструкцију и решавање потпроблема. Иако алгоритам не постиже на свим инстанцама најбоље резултате из литературе, показано је да се његове перформансе не разликују статистички значајно од резултата конкурентских метахеуристика, што потврђује његову практичнну вредност. Предложени алгоритам представља општи оквир који се може применити на широк спектар НП-тешких проблема. IRIS показује да комбинација егзактних и апроксимативних метода може бити валидна алтернатива за решавање доминацијских проблема на великим графовима.

Поглавље 6

Закључак

У овој дисертацији проучавана су четири доминацијска проблема на графовима: проблем минималне тежинске тоталне доминације, проблем минималне тежинске независне доминације, проблем k-јаке римске доминације, као и основни проблем минималне доминације, са посебним освртом на велике графове. Предложене су ефикасне методе за решавање сваког од ових проблема, укључујући хеуристичке и метахеуристичке алгоритме, егзактне методе засноване на целобројном линеарном програмирању, као и хибридни приступ који комбинује предности апроксимативних и егзактних метода. Посебан акценат стављен је на решавање доминацијских проблема на инстанцама великих димензија, где су традиционалне методе често неефикасне, а који се природно јављају у бројним практичним применама, као што су друштвене, телекомуникационе и биолошке мреже.

За решавање проблема минималне тежинске тоталне доминације, развијена је метода променљивих околина чије главне компоненте укључују размрдавање решења засновано на избацивању чворова, као и две врсте локалне претраге са првим побољшањем. Бржа верзија локалне претраге користи оператор 1-инверзије и извршава се у свакој итерацији алгоритма, док друга верзија користи оператор 1-замене и због квадратне временске сложености извршава се ређе. Функција прилагођености је пажљиво дизајнирана тако да омогућава процену квалитета како допустивих, тако и недопустивих решења. На овај начин, омогућен је суптилан пролазак кроз допустиве и недопустиве регионе простора претраге. Поред тога, имплементирано је инкрементално израчунавање вредности функције прилагођености, што значајно убрзава извршавање целокупног алгоритма. Експериментални резултати показују квалитет предложеног алгоритма. На готово свим малим и средњим инстанцама, достиже оптимална решења, док на великим инстанцама надмашује све конкурентске приступе, што је потврђено статистичким тестовима. Додатно, предложена је и примена на убрзавање ширења информација на друштвеним мрежама. Скуп иницијално информисаних чворова је постављен на решење проблема минималне тежинске тоталне доминације, а затим је за симулацију ширења информација коришћен SIR модел. Експерименталним резултатима показано је да оваква стратегија доводи до брже конвергенције у односу на насумични одабир иницијално информисаних чворова. Потенцијални правци будућих истраживања укључују теоријску анализу проблема на различитим класама графова, што би омогућило бољу иницијализацију и вођење претраге у будућим алгоритмима. Поред тога, комплексне мреже представљају плодан терен за примену овог и сличних проблема – њихова структура може инспирисати нове сценарије примене, па и потпуно нове варијанте проблема.

За решавање проблема минималне тежинске независне доминације, предложена су два модела целобројног линеарног програмирања (New-1 и New-2), као и похлепна хеуристика (Greedy-New). Спроведени су обимни експерименти ради процене квалитета развијених метода. Представљени ILP модели постижу оптимална решења на свим инстанцама са 100 чворова, а надмашују резултате конкурентских егзактних приступа и у већини преосталих случајева. Иако су метахеуристички приступи и даље пожељнији

за веће инстанце, предложени ILP модели представљају веома добру алтернативу за одређене класе инстанци, нарочито за случајне геометријске графове оријентисане на чворове. Значајно је истаћи да модел New-2 на графовима мање густине показује значајно краће време извршавања у односу на метахеуристике. Развијени похлепни алгоритам Greedy-New надмашује конкурентске похлепне хеуристике на већини инстанци. Поред тога, показује супериорне перформансе и у односу на неке конкурентске ILP моделе, поготово на случајним графовима са већим бројем чворова. Будућа истраживања могу ићи у правцу интеграције предложених метода као градивних блокова у хибридним или метахеуристичким приступима. Додатно, анализа понашања развијених метода на реалним графовима може пружити нове увиде и инспирацију за даља побољшања.

За проблем k-јаке римске доминације, чија се тежина огледа већ у самој провери допустивости решења која је експоненцијална у односу на број чворова графа, развијена су два приступа: похлепна хеуристика заснована на информацијама о покривености чворова, као и метода променљивих околина која користи поменути похлепни алгоритам за иницијализацију. VNS алгоритам је заснован на три кључне компоненте: размрдавање, квазидопустивост и локална претрага. Размдрдавањем се на случајан начин мења решење тако да укупан број јединица буде за један мањи, што може довести до недопустивог решења. Стога је задатак локалне претраге да без промене укупног броја јединица пронађе ново квазидопустиво решење. Како је већ поменуто, провера допустивости решења је експоненцијална, те је уведен концепт квазидопустивости, који омогућава ефикасну процену допустивости решења током претраге. Спроведени експерименти на случајним графовима, ад-хок бежичним мрежама различитих густина и реалним графовима добијеним из просторних (GeoJSON) података показују да предложени VNS значајно побољшава резултате похлепног алгоритма на готово свим инстанцама. Поред тога, на скоро свим инстанцама на којима су егзактне методе применљиве, VNS постиже оптимална или висококвалитетна решења у знатно краћем времену, а на великим инстанцама доследно надмашује конкурентске егзактне приступе, који имају проблеме са ограничењима меморије и времена. Практична примена алгоритма илустрована је кроз студију случаја оптималног позиционирања ватрогасних станица и возила у градским општинама, тако да читав град буде безбедан у случају k истовремених пожара. Потенцијални правци будућих истраживања обухватају развој скалабилнијих метода за графове са хиљадама чворова, што би могло укључивати декомпозицију графа коришћењем алгоритама за проналажење заједница, уз накнадну интеграцију појединачних решења. Додатно, могуће је размотрити развој модела машинског учења за апроксимативно рачунање вредности функције прилагођености, што би могло довести до убрзања у рачунски најскупљем делу алгоритма.

Разматран је и основни проблем минималне доминације са посебним фокусом на велике графове. Предложен је хибридни приступ IRIS који итеративно гради и решава пажљиво одабране потпроблеме коришћењем ILP решавача. У свакој итерацији, потпроблем се формира фиксирањем одређених променљивих на вредности из тренутног решења, док се преостале остављају слободним. На овај начин се комбинује снага егзактних метода са ефикасношћу метахеуристичких приступа. Алгоритам је дизајниран тако да буде применљив на широк спектар НП-тешких проблема, а у овом раду је прилагођен за проблем минималне доминације. Експериментални резултати показују да иако IRIS не постиже најбоље резултате у поређењу са најбољим метахеуристичким методама из литературе, не постоји статистички значајна разлика између њега и конкурентских приступа. Дакле, предложени алгоритам постиже конкурентне резултате, те представља валидну алтернативу за решавање проблема минималне доминације на великим графовима, као и, потенцијално, за друге сличне проблеме. Потенцијални правци будућих истраживања обухватају развој напреднијих стратегија за формирање потпроблема,

који могу укључивати примену алгоритама машинског учења ради побољшања избора променљивих које треба фиксирати. Поред тога, постоји потреба за побољшањима код изузетно великих графова, са милионима чворова, што може захтевати паралелизацију неких израчунавања, као и додатно смањивање величине потпроблема. С обзиром на то да је IRIS дизајниран као општи оквир, будућа истраживања могу укључивати примену овог приступа на друге НП-тешке проблеме код којих се ILP решавачи показују ефикасним.

Сумирајући доприносе и резултате представљене у овој дисертацији, може се закључити да је комбиновање хеуристичких, метахеуристичких и егзактних техника перспективан смер за решавање различитих доминацијских проблема на великим графовима. Указано је да су доминацијски проблеми богато и активно поље истраживања, као и да постоји значајан простор за даља побољшања и иновације. Принципи представљени у овом раду могу послужити као основа за развој скалабилних и ефикасних метода за решавање широког спектра НП-тешких проблема на комплексним мрежама.

Литература

[1] H Abdollahzadeh Ahangar и др. „Signed Roman domination in graphs". У: *Journal of Combinatorial Optimization* 27.2 (2014.), стр. 241–255.

[2] Salwani Abdullah, Edmund K Burke и Barry McCollum. „An investigation of variable neighbourhood search for university course timetabling". У: *The 2nd multidisciplinary international conference on scheduling: theory and applications (MISTA)*. 2005., стр. 413–427.

[3] Faisal N Abu-Khzam и др. „Feature Selection via Weighted Independent Domination". У: *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)*. IEEE. 2024., стр. 179–184.

[4] Hossein Abdollahzadeh Ahangar и др. „Total Roman domination in graphs". У: *Applicable Analysis and Discrete Mathematics* 10.2 (2016.), стр. 501–517.

[5] Mehmet Anıl Akbay, Albert López Serrano и Christian Blum. „A self-adaptive variant of CMSA: application to the minimum positive influence dominating set problem". У: *International Journal of Computational Intelligence Systems* 15.1 (2022.), стр. 44.

[6] Eduardo Álvarez-Miranda и Markus Sinnl. „Exact and heuristic algorithms for the weighted total domination problem". У: *Computers & Operations Research* 127 (2021.), стр. 105157.

[7] Guilherme Ferraz de Arruda, Francisco A Rodrigues и Yamir Moreno. „Fundamentals of spreading processes in single and multilayer complex networks". У: *Physics Reports* 756 (2018.), стр. 1–59.

[8] David A Bader и др. *Graph partitioning and graph clustering*. Sv. 588. American Mathematical Society Providence, RI, 2013.

[9] Balabhaskar Balasundaram и Sergiy Butenko. „Graph domination, coloring and cliques in telecommunications". У: *Handbook of optimization in telecommunications*. Springer, 2006., стр. 865–890.

[10] Cynthia Barnhart и др. „Branch-and-price: Column generation for solving huge integer programs". У: *Operations research* 46.3 (1998.), стр. 316–329.

[11] Joseph R Barr, Faisal N Abu-Khzam и Peter Shaw. „Feature selection via independent domination". У: *2023 Fifth International Conference on Transdisciplinary AI (TransAI)*. IEEE. 2023., стр. 197–200.

[12] Robert A Beeler, Teresa W Haynes и Stephen T Hedetniemi. „Double roman domination". У: *Discrete Applied Mathematics* 211 (2016.), стр. 23–29.

[13] Jacobus Franciscus Benders. „Partitioning procedures for solving mixed-variables programming problems". У: *Numer. Math* 4.1 (1962.), стр. 238–252.

[14] Claude Berge. „Théorie des graphes et ses applications". У: *(No Title)* (1958.).

[15] Alan A Bertossi. „Dominating sets for split and bipartite graphs". У: *Information processing letters* 19.1 (1984.), стр. 37–40.

[16] Hans-Georg Beyer и Hans-Paul Schwefel. „Evolution strategies–a comprehensive introduction". У: *Natural computing* 1.1 (2002.), стр. 3–52.

[17] Christian Blum и др. „Construct, merge, solve & adapt a new general algorithm for combinatorial optimization". У: *Computers & Operations Research* 68 (2016.), стр. 75–88.

[18] Christian Blum и др. „Hybrid metaheuristics in combinatorial optimization: A survey". У: *Applied soft computing* 11.6 (2011.), стр. 4135–4151.

[19] Kellogg S Booth и J Howard Johnson. „Dominating sets in chordal graphs". У: *SIAM Journal on Computing* 11.1 (1982.), стр. 191–199.

[20] Salim Bouamama и Christian Blum. „An improved greedy heuristic for the minimum positive influence dominating set problem in social networks". У: *Algorithms* 14.3 (2021.), стр. 79.

[21] Salim Bouamama, Christian Blum и Jean-Guillaume Fages. „An algorithm based on ant colony optimization for the minimum connected dominating set problem". У: *Applied Soft Computing* 80 (2019.), стр. 672–686.

[22] David Chalupa. „An order-based algorithm for minimum dominating set with application in graph mining". У: *Information Sciences* 426 (2018.), стр. 101–116.

[23] Shun-Chieh Chang, Jia-Jie Liu и Yue-Li Wang. „The weighted independent domination problem in series-parallel graphs". У: *Intelligent Systems and Applications*. IOS Press, 2015., стр. 77–84.

[24] Antoine Charpentier и др. „Variable neighborhood search with cost function networks to solve large computational protein design problems". У: *Journal of Chemical Information and Modeling* 59.1 (2018.), стр. 127–136.

[25] Mainak Chatterjee, Sajal K Das и Damla Turgut. „WCA: A weighted clustering algorithm for mobile ad hoc networks". У: *Cluster computing* 5 (2002.), стр. 193–204.

[26] Mustapha Chellali и др. „Roman domination in graphs". У: *Topics in domination in graphs* (2020.), стр. 365–409.

[27] Mustapha Chellali и др. „Varieties of Roman domination IV". У: *AKCE International Journal of Graphs and Combinatorics* (2025.), стр. 1–30.

[28] Miroslav Chlebík и Janka Chlebíková. „Approximation hardness of dominating set problems". У: *European Symposium on Algorithms*. Springer. 2004., стр. 192–203.

[29] Brent N Clark, Charles J Colbourn и David S Johnson. „Unit disk graphs". У: *Discrete mathematics* 86.1-3 (1990.), стр. 165–177.

[30] Ernest J Cockayne, RM Dawes и Stephen T Hedetniemi. „Total domination in graphs". У: *Networks* 10.3 (1980.), стр. 211–219.

[31] Ernest J Cockayne и Stephen T Hedetniemi. „Towards a theory of domination in graphs". У: *Networks* 7.3 (1977.), стр. 247–261.

[32] Ernie J Cockayne и др. „Roman domination in graphs". У: *Discrete mathematics* 278.1-3 (2004.), стр. 11–22.

[33] Mohammad Mehdi Daliri Khomami и др. „Minimum positive influence dominating set and its application in influence maximization: a learning automata approach". У: *Applied Intelligence* 48 (2018.), стр. 570–593.

[34] Mirela Damian и Sriram V Pemmaraju. „APX-hardness of domination problems in circle graphs". У: *Information processing letters* 97.6 (2006.), стр. 231–237.

[35] George B Dantzig. „Reminiscences about the origins of linear programming". У: *Mathematical Programming The State of the Art: Bonn 1982*. Springer, 1983., стр. 78–86.

[36] George B Dantzig, Alex Orden, Philip Wolfe и др. „The generalized simplex method for minimizing a linear form under linear inequality restraints". У: *Pacific Journal of Mathematics* 5.2 (1955.), стр. 183–195.

[37] Pedro Pinacho Davidson, Christian Blum и Jose A Lozano. „The weighted independent domination problem: Integer linear programming models and metaheuristic approaches". У: *European Journal of Operational Research* 265.3 (2018.), стр. 860–871.

[38] Jacques Desrosiers и Marco E Lübbecke. „Branch-price-and-cut algorithms". У: *Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Chichester* (2011.), стр. 109–131.

[39] Marko Djukanović и др. „Graph protection under multiple simultaneous attacks: A heuristic approach". У: *Knowledge-Based Systems* 309 (2025.), стр. 112791.

[40] Marco Dorigo, Mauro Birattari и Thomas Stutzle. „Ant colony optimization". У: *IEEE computational intelligence magazine* 1.4 (2007.), стр. 28–39.

[41] Rodney G Downey и др. „Parameterized approximation of dominating set problems". У: *Information Processing Letters* 109.1 (2008.), стр. 68–70.

[42] Paul Erdős и Alfréd Rényi. „On random graphs I". У: *Publ. math. debrecen* 6.290-297 (1959.), стр. 18.

[43] Thomas A Feo и Mauricio GC Resende. „Greedy randomized adaptive search procedures". У: *Journal of global optimization* 6.2 (1995.), стр. 109–133.

[44] Yaacov Fernandess и Dahlia Malkhi. „K-clustering in wireless ad hoc networks". У: *Proceedings of the second ACM international workshop on Principles of mobile computing*. 2002., стр. 31–37.

[45] Vladimir Filipović и Aleksandar Kartelj. „Topological variable neighborhood search". У: *Journal of Big Data* 11.1 (2024.), стр. 178.

[46] Matteo Fischetti, Ivana Ljubić и Markus Sinnl. „Redesigning Benders decomposition for large-scale facility location". У: *Management Science* 63.7 (2017.), стр. 2146–2162.

[47] Milton Friedman. „The use of ranks to avoid the assumption of normality implicit in the analysis of variance". У: *Journal of the american statistical association* 32.200 (1937.), стр. 675–701.

[48] Michel Gendreau. „An Introduction to Tabu Search". У: *Handbook of Metaheuristics*. Urednik Fred Glover и Gary A. Kochenberger. Boston, MA: Springer US, 2003., стр. 37–54. ISBN: 978-0-306-48056-0. DOI: 10.1007/0-306-48056-5_2. URL: https://doi.org/10.1007/0-306-48056-5_2.

[49] Michel Gendreau и Jean-Yves Potvin, ур. *Handbook of Metaheuristics*. 3rd. Sv. 272. International Series in Operations Research & Management Science. Springer, 2019. ISBN: 978-3-319-91086-3. DOI: 10.1007/978-3-319-91086-3.

[50] Fred Glover. „Future paths for integer programming and links to artificial intelligence". У: *Computers & operations research* 13.5 (1986.), стр. 533–549.

[51] Fabrizio Grandoni. „A note on the complexity of minimum dominating set". У: *Journal of Discrete Algorithms* 4.2 (2006.), стр. 209–214.

[52] Milana Grbić и др. „Variable Neighborhood Search for Partitioning Sparse Biological Networks into the Maximum Edge-Weighted k k-Plexes". У: *IEEE/ACM transactions on computational biology and bioinformatics* 17.5 (2019.), стр. 1822–1831.

[53] Daniel A Griffith и Ulemu Luhanga. „Approximating the inertia of the adjacency matrix of a connected planar graph that is the dual of a geographic surface partitioning". У: *Geographical Analysis* 43.4 (2011.), стр. 383–402.

[54] Preeti Gupta. „Domination in graph with application". У: *Indian J. Res* 2.3 (2013.), стр. 115–117.

[55] Pierre Hansen, Nenad Mladenović и Jose A Moreno Perez. „Variable neighbourhood search: methods and applications". У: *Annals of Operations Research* 175 (2010.), стр. 367–407.

[56] Pierre Hansen, Nenad Mladenović и Dragan Urošević. „Variable neighborhood search for the maximum clique". У: *Discrete Applied Mathematics* 145.1 (2004.), стр. 117–125.

[57] Pierre Hansen, Ceyda Oğuz и Nenad Mladenović. „Variable neighborhood search for minimum cost berth allocation". У: *European journal of operational research* 191.3 (2008.), стр. 636–649.

[58] Pierre Hansen и др. *Variable neighborhood search*. Springer, 2019.

[59] Teresa W Haynes, Stephen Hedetniemi и Peter Slater. *Fundamentals of domination in graphs*. CRC press, 2013.

[60] Vera C Hemmelmayr, Karl F Doerner и Richard F Hartl. „A variable neighborhood search heuristic for periodic routing problems". У: *European Journal of Operational Research* 195.3 (2009.), стр. 791–802.

[61] John Henry Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975. ISBN: 9780472084609. URL: `https://books.google.rs/books?id=YE5RAAAAMAAJ`.

[62] Shuli Hu и др. „Towards efficient local search for the minimum total dominating set problem". У: *Applied Intelligence* 51.12 (2021.), стр. 8753–8767.

[63] Marija Ivanović и Dragan Urošević. „Variable Neighborhood Search Approach for Solving Roman and Weak Roman Domination Problems on Graphs." У: *Computing & Informatics* 38.1 (2019.).

[64] Carl Friedrich Jaenisch. *Traité des applications de l'analyse mathématique au jeu des échecs: précédé d'une introduction à l'usage des lecteurs, soit étrangers aux échecs, soit peu versés dans l'analyse*. Sv. 1. Londres, 1862.

[65] Bassem Jarboui и др. „Variable neighborhood search for location routing". У: *Computers & Operations Research* 40.1 (2013.), стр. 47–57.

[66] Leonid V Kantorovich. „Mathematical methods of organizing and planning production". У: *Management science* 6.4 (1960.), стр. 366–422.

[67] Stefan Kapunac. „Integer Linear Programming Models and Greedy Heuristic for the Minimum Weighted Independent Dominating Set Problem". У: *Serdica Journal of Computing* 17.2 (2023.), стр. 117–136.

[68] Stefan Kapunac, Aleksandar Kartelj и Marko Djukanović. „Variable neighborhood search for weighted total domination problem and its application in social network information spreading". У: *Applied Soft Computing* 143 (2023.), стр. 110387.

[69] Narendra Karmarkar. „A new polynomial-time algorithm for linear programming". У: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. 1984., стр. 302–311.

[70] Richard M Karp. „Reducibility among combinatorial problems". У: *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*. Springer, 2009., стр. 219–241.

[71] James Kennedy и Russell Eberhart. „Particle swarm optimization". У: *Proceedings of ICNN'95-international conference on neural networks*. Sv. 4. ieee. 1995., стр. 1942–1948.

[72] Leonid G Khachiyan. „Polynomial algorithms in linear programming". У: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980.), стр. 53–72.

[73] Pooja Khurana и Deepak Kumar. „Sir model for fake news spreading through whatsapp". У: *Proceedings of 3rd international conference on Internet of Things and connected technologies (ICIoTCT)*. 2018., стр. 26–27.

[74] Yosuke Kikuchi и Yukio Shibata. „On the domination numbers of generalized de Bruijn digraphs and generalized Kautz digraphs". У: *Information Processing Letters* 86.2 (2003.), стр. 79–85.

[75] Scott Kirkpatrick, C Daniel Gelatt Jr и Mario P Vecchi. „Optimization by simulated annealing". У: *science* 220.4598 (1983.), стр. 671–680.

[76] Victor Klee и George J Minty. „How good is the simplex algorithm". У: *Inequalities* 3.3 (1972.), стр. 159–175.

[77] Ailsa H Land и Alison G Doig. „An automatic method for solving discrete programming problems". У: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer, 2009., стр. 105–132.

[78] Jure Leskovec. „SNAP Datasets: Stanford large network dataset collection". У: *Retrieved December 2021 from http://snap. stanford. edu/data* (2014.).

[79] Jure Leskovec, Lada A Adamic и Bernardo A Huberman. „The dynamics of viral marketing". У: *ACM Transactions on the Web (TWEB)* 1.1 (2007.), 5–es.

[80] Chuang Liu и Zi-Ke Zhang. „Information spreading on dynamic social networks". У: *Communications in Nonlinear Science and Numerical Simulation* 19.4 (2014.), стр. 896–904.

[81] Chunmei Liu и Yinglei Song. „Parameterized complexity and inapproximability of dominating set problem in chordal and near chordal graphs". У: *Journal of combinatorial optimization* 22.4 (2011.), стр. 684–698.

[82] Jie Liu и др. „Analysis of rumor spreading in communities based on modified SIR model in microblog". У: *Artificial Intelligence: Methodology, Systems, and Applications: 16th International Conference, AIMSA 2014, Varna, Bulgaria, September 11-13, 2014. Proceedings 16*. Springer. 2014., стр. 69–79.

[83] Yang-Yu Liu, Jean-Jacques Slotine и Albert-László Barabási. „Controllability of complex networks". У: *nature* 473.7346 (2011.), стр. 167–173.

[84] Zeyu Liu, Xueping Li и Anahita Khojandi. „On the k-strong Roman domination problem". У: *Discrete Applied Mathematics* 285 (2020.), стр. 227–241.

[85] Manuel López-Ibáñez и др. „The irace package: Iterated racing for automatic algorithm configuration". У: *Operations Research Perspectives* 3 (2016.), стр. 43–58.

[86] Helena R Lourenço, Olivier C Martin и Thomas Stützle. „Iterated local search". У: *Handbook of metaheuristics*. Springer, 2003., стр. 320–353.

[87] Yuede Ma, Qingqiong Cai и Shunyu Yao. „Integer linear programming models for the weighted total domination problem". У: *Applied Mathematics and Computation* 358 (2019.), стр. 146–150.

[88] Nenad Mladenović и Pierre Hansen. „Variable neighborhood search". У: *Computers & operations research* 24.11 (1997.), стр. 1097–1100.

[89] Nenad Mladenović и др. „Variable neighborhood search for metric dimension and minimal doubly resolving set problems". У: *European Journal of Operational Research* 220.2 (2012.), стр. 328–337.

[90] Yamir Moreno, Maziar Nekovee и Amalio F Pacheco. „Dynamics of rumor spreading in complex networks". У: *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 69.6 (2004.), стр. 066130.

[91] Jose C Nacher и Tatsuya Akutsu. „Dominating scale-free networks with variable scaling exponent: heterogeneous networks are not difficult to control". У: *New Journal of Physics* 14.7 (2012.), стр. 073005.

[92] Mallikarjun Rao Nakkala, Alok Singh и André Rossi. „Multi-start iterated local search, exact and matheuristic approaches for minimum capacitated dominating set problem". У: *Applied Soft Computing* 108 (2021.), стр. 107437.

[93] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

[94] J Nieminen. „Two bounds for the domination number of a graph". У: *IMA Journal of Applied Mathematics* 14.2 (1974.), стр. 183–187.

[95] Bojan Nikolić и др. „Theoretical studies of the k-strong Roman domination problem". У: *Kuwait Journal of Science* 51.4 (2024.), стр. 100283.

[96] Jorge Nocedal и Stephen J Wright. *Numerical optimization.* Springer, 2006.

[97] O. Ore. *Theory of Graphs.* American Mathematical Society colloquium publications. American Math. Soc., 1962. ISBN: 9780821874714. URL: `https://books.google.rs/books?id=z6zLw18DyuoC`.

[98] Shiwei Pan и др. „An improved master-apprentice evolutionary algorithm for minimum independent dominating set problem". У: *Frontiers of Computer Science* 17.4 (2023.), стр. 174326.

[99] Yexin Peng и др. „A dual-evaluation-mode local-search for the minimum dominating set problem on ultra-large sparse graphs: Y. Peng et al." У: *The Journal of Supercomputing* 81.13 (2025.), стр. 1244.

[100] Milan Predojević, Aleksandar Kartelj и Marko Djukanović. „Variable neighborhood search for solving the k-domination problem". У: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation.* 2023., стр. 239–242.

[101] Yongsheng Rao и др. „A survey on domination in vague graphs with application in transferring cancer patients between countries". У: *Mathematics* 9.11 (2021.), стр. 1258.

[102] Ingo Rechenberg. „Evolutionsstrategien". У: *Simulationsmethoden in der Medizin und Biologie: Workshop, Hannover, 29. Sept.–1. Okt. 1977.* Springer. 1978., стр. 83–114.

[103] Helena Sofia Rodrigues и Manuel José Fonseca. „Can information be spread as a virus? Viral marketing as epidemiological model". У: *Mathematical methods in the applied sciences* 39.16 (2016.), стр. 4780–4786.

[104] Vahid Roshanaei и др. „A variable neighborhood search for job shop scheduling with set-up times to minimize makespan". У: *Future Generation Computer Systems* 25.6 (2009.), стр. 654–661.

[105] Le Shu, Tianyang Ma и Longin Jan Latecki. „Stable feature selection with minimal independent dominating sets". У: *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*. 2013., стр. 450–457.

[106] Gerard Sierksma и Yori Zwols. *Linear and integer optimization: theory and practice*. CRC Press, 2015.

[107] Jialin Song, Yisong Yue, Bistra Dilkina и др. „A general large neighborhood search framework for solving integer linear programs". У: *Advances in Neural Information Processing Systems* 33 (2020.), стр. 20012–20023.

[108] Rainer Storn и Kenneth Price. „Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". У: *Journal of global optimization* 11.4 (1997.), стр. 341–359.

[109] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley, 2009. ISBN: 978-0-470-27858-1.

[110] Yiyuan Wang и др. „A local search algorithm with reinforcement learning based repair procedure for minimum weight independent dominating set". У: *Information Sciences* 512 (2020.), стр. 533–548.

[111] Yiyuan Wang и др. „A path cost-based GRASP for minimum independent dominating set problem". У: *Neural Computing and Applications* 28 (2017.), стр. 143–151.

[112] Laurence A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998. ISBN: 9780471283669. URL: `https://books.google.rs/books?id=x7RvQgAACAAJ`.

[113] Stefan Wuchty. „Controllability in protein interaction networks". У: *Proceedings of the National Academy of Sciences* 111.19 (2014.), стр. 7156–7160.

[114] Xin-Fang Yan, Ai-Qin Liu и Ting Yang. „A virtual backbone network algorithm based on a minimal independent dominating set for manets". У: *Dianzi Xuebao(Acta Electronica Sinica)* 35.6 (2007.), стр. 1134–1138.

[115] Jiguo Yu и др. „Connected dominating sets in wireless ad hoc and sensor networks–A comprehensive survey". У: *Computer Communications* 36.2 (2013.), стр. 121–134.

[116] Yongli Zan и др. „SICR rumor spreading model in complex networks: Counterattack and self-resistance". У: *Physica A: Statistical Mechanics and its Applications* 405 (2014.), стр. 159–170.

[117] Enqiang Zhu и др. „A dual-mode local search algorithm for solving the minimum dominating set problem". У: *Knowledge-Based Systems* 298 (2024.), стр. 111950.

Додатак А

Детаљи имплементације инкременталног рачунања вредности функције прилагођености

Алгоритам 13 Функција recalcNodeAdded

Улаз: S: решење; $G = (V, Ef_1, f_2)$: инстанца проблема; v: чвор за додавање; $viol$: број чворова без суседа у S; $objValue$: вредност функције циља S; $external$: листа сортираних (по тежини грана) скупова екстерних грана.
$viol_{new} \leftarrow viol$
$objValue_{new} \leftarrow objValue + f_1(v)$ ▷ додавање тежине чвора
if $external[v] \neq \emptyset$ **then** ▷ v је имао суседа у S
    $objValue_{new} \leftarrow objValue_{new} - f_2(external[v][0])$
end if
for $e \in external[v]$ **do** ▷ додавање тежине интерних грана
    $objValue_{new} \leftarrow objValue_{new} + f_2(e)$
end for
for $e = (v, v') \in external[v]$ **do**
    $weight \leftarrow f_2(e)$
    if $external[v'] \neq \emptyset$ **then**
        $viol_{new} \leftarrow viol_{new} - 1$
        if $v' \notin S$ **then** ▷ интерне гране су већ додате, додају се само екстерне
            $objValue_{new} \leftarrow objValue_{new} + weight$
        end if
    else
        $prev_{min\_weight} \leftarrow f_2(external[v'][0])$
        if $v' \notin \bar{S}$ and $prev_{min\_weight} < weight$ **then** ▷ v је нови најближи сусед v' у S
            $objValue_{new} \leftarrow objValue_{new} - prev_{min\_weight} + weight$
        end if
    end if
end for
Излаз: $viol_{new} + \frac{objValue_{new}}{W_{tot}+1}$

Алгоритам 14 Функција recalcNodeRemoved

Улаз: S: решење; $G = (V, E, f_1, f_2)$: инстанца проблема; v: чвор за избацивање; $viol$: број чворова без суседа у S; $objValue$: вредност функције циља S; $external$: листа сортираних (по тежини грана) скупова екстерних грана.
$viol_{new} \leftarrow viol$
$objValue_{new} \leftarrow objValue - f_1(v)$ ▷ одузимање тежине чвора
if $external[v] \neq \emptyset$ **then** ▷ ако је чвор имао суседа у S, додавање тежине његове екстерне гране
    $objValue_{new} \leftarrow objValue_{new} + f_2(external[v][0])$
end if
for $e \in external[v]$ **do** ▷ одузимање тежине интерних грана
    $objValue_{new} \leftarrow objValue_{new} - f_2(edge)$
end for
for $e = (v, v') \in external[v]$ **do**
    $weight \leftarrow f_2(e)$
    if $|external[v']| = 1$ **then** ▷ кардиналност скупа је 1
        $viol_{new} \leftarrow viol_{new} + 1$
        if $v' \notin s$ **then** ▷ интерне гране су већ одузете, сада се одузимају само екстерне
            $objValue_{new} \leftarrow objValue_{new} - weight$
        end if
    else
        $prev_{min\_edge} = (v', u) \leftarrow external[v'][0]$
        if $v' \notin S$ and $u = v$ **then** ▷ v више није најближи сусед v' у S
            $objValue_{new} \leftarrow objValue_{new} - f_2(prev_{min\_edge}) + f_2(external[v'][1])$
        end if
    end if
end for
Излаз: $viol_{new} + \frac{objValue_{new}}{W_{tot}+1}$

Додатак Б

Додатни резултати за MWTDS проблем

У табели Б.1 приказани су детаљни резултати поређене четири методе за инстанце из скупа `NEW` са 250 чворова. Из ње се могу извући следећи закључци:

- Ilp не успева оптимално да реши ниједну инстанцу услед достизања временског ограничења. Поред тога, не проналази ниједно најбоље познато решење.
- Vns проналази најбоље познато решење на 40 од 45 инстанци, док Grasp+Ga постиже најбоље решење на 29 инстанци. Ово указује на бољу скалабилност Vns-а.
- Као што је и очекивано, Grasp метода је најбржа, али и најслабија у погледу квалитета добијених решења.

Табела Б.1: Детаљно поређење Vns-а у односу на Grasp и Grasp+Ga на скупу NEW-250

		Ilp		Vns				Grasp				Grasp+Ga			
инстанца	$\overline{best}$	$\overline{obj}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*
NEW-250-0.2-10-50-1	1662	1933	TL	111.2	1703	2.47		10.5	1871	12.58		122.4	**1662**	0	best
NEW-250-0.2-10-50-2	1728	2029	TL	108.4	**1728**	0	best	10.4	1943	12.44		125.4	1768	2.31	
NEW-250-0.2-10-50-3	1754	1943	TL	96.6	1769	0.86		10.3	1957	11.57		143.6	**1754**	0	best
NEW-250-0.2-10-50-4	1635	1799	TL	100.8	**1635**	0	best	10.2	1796	9.85		121	1655	1.22	
NEW-250-0.2-10-50-5	1737	1905	TL	103.6	**1737**	0	best	10.6	1924	10.77		135.6	1739	0.12	
NEW-250-0.2-25-25-1	1144	1244	TL	102.7	**1144**	0	best	10.4	1300	13.64		141.5	1157	1.14	
NEW-250-0.2-25-25-2	1135	1210	TL	101.9	**1135**	0	best	10	1331	17.27		112.9	1139	0.35	
NEW-250-0.2-25-25-3	1132	1197	TL	109.2	**1132**	0	best	10.6	1314	16.08		120.1	**1132**	0	best
NEW-250-0.2-25-25-4	1162	1228	TL	107.7	**1162**	0	best	10.4	1227	5.59		103.5	**1162**	0	best
NEW-250-0.2-25-25-5	1147	1285	TL	111.3	**1147**	0	best	10.3	1306	13.86		124.2	1149	0.17	
NEW-250-0.2-50-10-1	749	763	TL	108.2	**749**	0	best	10.8	806	7.61		101.8	**749**	0	best
NEW-250-0.2-50-10-2	708	715	TL	102.4	**708**	0	best	10.1	754	6.5		110.4	**708**	0	best
NEW-250-0.2-50-10-3	719	720	TL	104.3	**719**	0	best	10.2	756	5.15		103.7	**719**	0	best
NEW-250-0.2-50-10-4	680	**680**	TL	117.9	**680**	0	best	10	723	6.32		107.7	**680**	0	best
NEW-250-0.2-50-10-5	758	765	TL	105.9	**758**	0	best	10	788	3.96		127.9	764	0.79	
NEW-250-0.5-10-50-1	1431	1696	TL	195.2	**1431**	0	best	24.1	1536	7.34		175.7	**1431**	0	best
NEW-250-0.5-10-50-2	1468	1834	TL	223.5	**1468**	0	best	24.1	1622	10.49		168.5	1471	0.2	
NEW-250-0.5-10-50-3	1417	1502	TL	234.8	**1417**	0	best	24	1579	11.43		205.7	**1417**	0	best
NEW-250-0.5-10-50-4	1492	1716	TL	208.8	**1492**	0	best	24.7	1632	9.38		183.2	1518	1.74	
NEW-250-0.5-10-50-5	1474	1798	TL	205	1483	0.61		24.3	1549	5.09		175.6	**1474**	0	best
NEW-250-0.5-25-25-1	900	1018	TL	204.4	**900**	0	best	24.2	981	9		182	914	1.56	
NEW-250-0.5-25-25-2	921	1044	TL	195.4	**921**	0	best	24.1	1005	9.12		171.9	939	1.95	
NEW-250-0.5-25-25-3	896	1100	TL	195	**896**	0	best	25.6	977	9.04		164.4	**896**	0	best
NEW-250-0.5-25-25-4	956	1082	TL	237.6	**956**	0	best	26	1017	6.38		189.5	957	0.1	
NEW-250-0.5-25-25-5	914	1060	TL	219.4	915	0.11		25.7	996	8.97		210.2	**914**	0	best
NEW-250-0.5-50-10-1	533	535	TL	188.1	**533**	0	best	25.2	561	5.25		161.8	**533**	0	best
NEW-250-0.5-50-10-2	554	588	TL	203.2	**554**	0	best	25	595	7.4		189	**554**	0	best
NEW-250-0.5-50-10-3	556	557	TL	173.7	**556**	0	best	24.9	571	2.7		166.8	**556**	0	best
NEW-250-0.5-50-10-4	558	569	TL	181.7	**558**	0	best	25.2	**558**	0	best	206.3	**558**	0	best
NEW-250-0.5-50-10-5	532	**532**	TL	186.1	**532**	0	best	24.6	562	5.64		143.6	**532**	0	best
NEW-250-0.8-10-50-1	1410	1627	TL	304.7	**1410**	0	best	42.4	1509	7.02		275.6	1435	1.77	
NEW-250-0.8-10-50-2	1401	1547	TL	275.3	**1401**	0	best	44.1	1467	4.71		272.1	**1401**	0	best
NEW-250-0.8-10-50-3	1444	1748	TL	313.8	**1444**	0	best	43.5	1518	5.12		313.1	**1444**	0	best
NEW-250-0.8-10-50-4	1420	1738	TL	308.7	**1420**	0	best	43.5	1441	1.48		294	**1420**	0	best
NEW-250-0.8-10-50-5	1430	1668	TL	290	**1430**	0	best	42.5	1520	6.29		269.1	**1430**	0	best
NEW-250-0.8-25-25-1	919	1006	TL	295.2	**919**	0	best	41.6	983	6.96		295.7	928	0.98	
NEW-250-0.8-25-25-2	835	931	TL	293.3	**835**	0	best	42.5	887	6.23		286.4	**835**	0	best
NEW-250-0.8-25-25-3	914	1072	TL	271.5	919	0.55		41.4	966	5.69		295.3	**914**	0	best
NEW-250-0.8-25-25-4	846	903	TL	289.6	**846**	0	best	41.3	893	5.56		268.1	**846**	0	best
NEW-250-0.8-25-25-5	853	911	TL	285.6	**853**	0	best	40.9	926	8.56		280.1	**853**	0	best
NEW-250-0.8-50-10-1	489	490	TL	274.8	**489**	0	best	41	496	1.43		295.3	**489**	0	best
NEW-250-0.8-50-10-2	507	522	TL	271.5	**507**	0	best	41.8	518	2.17		254.1	**507**	0	best
NEW-250-0.8-50-10-3	498	507	TL	274.8	**498**	0	best	41.6	513	3.01		323.9	499	0.2	
NEW-250-0.8-50-10-4	489	519	TL	289	**489**	0	best	43.2	496	1.43		287.1	494	1.02	
NEW-250-0.8-50-10-5	482	508	TL	253	**482**	0	best	42.6	518	7.47		339.3	**482**	0	best

Табела Б.2 приказује резултате за инстанце из скупа NEW са 500 чворова. Из ње се може закључити следеће:

- Ilp је у свим случајевима достигла временско ограничење и није пронашла ниједно најбоље решење.
- Vns постиже најбоље познато решење на 33 инстанце, док Grasp+Ga постиже најбоље решење на 24 инстанце.
- Методе Vns и Grasp+Ga су упоредиве на ређим графовима ($p = 0.2$), али на гушћим графовима Vns значајно доминира у погледу квалитета решења. Ово би се могло приписати размрдавању заснованом на избацивању чворова, које боље одговара гушћим графовима, где се очекује да висококвалитетно решење садржи мање чворова (сублинеарно у односу на укупан број чворова $|V|$).

Табела Б.2: Детаљно поређење Vns-а у односу на Grasp и Grasp+Ga на скупу NEW-500

		Ilp		Vns				Grasp				Grasp+Ga			
инстанца	$\overline{best}$	$\overline{obj}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*	$\overline{t}$	$\overline{obj}$	$\overline{pg\%}$	*ind.*
NEW-500-0.2-10-50-1	2893	3837	TL	650.8	2968	2.59		89.6	3304	14.21		728	**2893**	0	best
NEW-500-0.2-10-50-2	2937	3736	TL	767	2947	0.34		88.8	3254	10.79		806.5	**2937**	0	best
NEW-500-0.2-10-50-3	3082	3839	TL	737.2	**3082**	0	best	87.9	3425	11.13		940.2	3090	0.26	
NEW-500-0.2-10-50-4	2964	3705	TL	584.9	3063	3.34		88.9	3360	13.36		904.5	**2964**	0	best
NEW-500-0.2-10-50-5	2928	3965	TL	574.7	**2928**	0	best	88.7	3197	9.19		1069.2	2942	0.48	
NEW-500-0.2-25-25-1	1884	2383	TL	835.2	**1884**	0	best	87.1	2130	13.06		667.7	1918	1.8	
NEW-500-0.2-25-25-2	1931	2474	TL	603.1	1941	0.52		86.8	2125	10.05		803.8	**1931**	0	best
NEW-500-0.2-25-25-3	1898	2225	TL	664.5	**1898**	0	best	86.1	2064	8.75		913.9	1914	0.84	
NEW-500-0.2-25-25-4	1861	2368	TL	603.7	1937	4.08		87.1	2081	11.82		774.7	**1861**	0	best
NEW-500-0.2-25-25-5	1861	2285	TL	875.1	**1861**	0	best	86.8	2091	12.36		926.2	1904	2.31	
NEW-500-0.2-50-10-1	1128	1256	TL	667.7	**1128**	0	best	87.6	1209	7.18		853.2	1133	0.44	
NEW-500-0.2-50-10-2	1143	1263	TL	719.9	**1143**	0	best	88.5	1217	6.47		831.2	1159	1.4	
NEW-500-0.2-50-10-3	1108	1288	TL	633.6	**1108**	0	best	89.4	1182	6.68		627.9	**1108**	0	best
NEW-500-0.2-50-10-4	1145	1246	TL	715.8	**1145**	0	best	86.4	1222	6.72		708.2	1149	0.35	
NEW-500-0.2-50-10-5	1186	1301	TL	747.8	1188	0.17		87.1	1221	2.95		821.3	**1186**	0	best
NEW-500-0.5-10-50-1	2590	3864	TL	1560.1	**2590**	0	best	231.1	2788	7.64		1031.5	2614	0.93	
NEW-500-0.5-10-50-2	2542	4551	TL	1624.3	**2542**	0	best	230.2	2825	11.13		1089.1	2631	3.5	
NEW-500-0.5-10-50-3	2547	4759	TL	1432.8	**2547**	0	best	229.8	2769	8.72		1011.5	2552	0.2	
NEW-500-0.5-10-50-4	2584	3220	TL	1140.7	2596	0.46		230.4	2805	8.55		1112.4	**2584**	0	best
NEW-500-0.5-10-50-5	2539	3691	TL	821.4	**2539**	0	best	229.2	2858	12.56		978.2	**2539**	0	best
NEW-500-0.5-25-25-1	1543	2164	TL	1059.9	**1543**	0	best	228.8	1670	8.23		1064.9	1566	1.49	
NEW-500-0.5-25-25-2	1574	2137	TL	1069.6	1594	1.27		229.9	1750	11.18		1266.9	**1574**	0	best
NEW-500-0.5-25-25-3	1547	2121	TL	1093.1	**1547**	0	best	235.4	1696	9.63		1054.7	1568	1.36	
NEW-500-0.5-25-25-4	1564	2210	TL	986.5	1566	0.13		246.5	1667	6.59		1029.5	**1564**	0	best
NEW-500-0.5-25-25-5	1567	2343	TL	998.6	1570	0.19		239.9	1726	10.15		1083.5	**1567**	0	best
NEW-500-0.5-50-10-1	919	1036	TL	785.7	**919**	0	best	238.2	943	2.61		961.3	**919**	0	best
NEW-500-0.5-50-10-2	911	1067	TL	761	**911**	0	best	232.4	950	4.28		874.7	933	2.41	
NEW-500-0.5-50-10-3	906	1020	TL	808.5	**906**	0	best	235.7	971	7.17		850.5	**906**	0	best
NEW-500-0.5-50-10-4	904	1091	TL	852	**904**	0	best	243.9	949	4.98		995.5	**904**	0	best
NEW-500-0.5-50-10-5	932	1063	TL	986.2	**932**	0	best	236.5	956	2.58		945.8	938	0.64	
NEW-500-0.8-10-50-1	2500	3385	TL	1466	**2500**	0	best	403.8	2644	5.76		1479.9	**2500**	0	best
NEW-500-0.8-10-50-2	2497	3492	TL	1631.9	2499	0.08		408.6	2565	2.72		1524.7	**2497**	0	best
NEW-500-0.8-10-50-3	2466	3384	TL	1588.3	**2466**	0	best	401	2612	5.92		1729.6	2513	1.91	
NEW-500-0.8-10-50-4	2511	5055	TL	1552.9	**2511**	0	best	402.1	2667	6.21		1769.7	2544	1.31	
NEW-500-0.8-10-50-5	2497	4581	TL	1450.3	**2497**	0	best	407.5	2578	3.24		1790.4	2538	1.64	
NEW-500-0.8-25-25-1	1490	2004	TL	1326.6	**1490**	0	best	414.4	1616	8.46		1635.6	**1490**	0	best
NEW-500-0.8-25-25-2	1505	1972	TL	1523.8	**1505**	0	best	400.9	1605	6.64		1394.3	1509	0.27	
NEW-500-0.8-25-25-3	1470	2048	TL	1332.7	1471	0.07		401.6	1575	7.14		1693.8	**1470**	0	best
NEW-500-0.8-25-25-4	1489	1986	TL	1369.6	**1489**	0	best	390.8	1551	4.16		1532.8	**1489**	0	best
NEW-500-0.8-25-25-5	1496	2055	TL	1534	**1496**	0	best	391.9	1595	6.62		1469.2	1507	0.74	
NEW-500-0.8-50-10-1	871	1021	TL	1411.7	**871**	0	best	398.5	916	5.17		1368.9	**871**	0	best
NEW-500-0.8-50-10-2	853	968	TL	1406.6	**853**	0	best	398.2	886	3.87		1611.8	**853**	0	best
NEW-500-0.8-50-10-3	855	1002	TL	1426.6	**855**	0	best	389.3	873	2.11		1607	**855**	0	best
NEW-500-0.8-50-10-4	869	985	TL	1254.7	**869**	0	best	388.8	894	2.88		1510.7	871	0.23	
NEW-500-0.8-50-10-5	872	933	TL	1421.1	**872**	0	best	390.7	892	2.29		1313.8	**872**	0	best

Додатак B

Детаљи имплементације детерминистичке и стохастичке стратегије одбране

Алгоритам 15 Функција `deterministicDefense`

Улаз: решење s, граф $G = (V, E), k, allAlternatives, reducedAttack, defendingNodes$
Излаз: пар *attackDefended*, *defendingNodes*

```
1: for alternative ∈ allAlternatives do
2:     attackDefended ← True
3:     for v ∈ reducedAttack do
4:         u ← alternative[v]
5:         if canHelp(u, v) then
6:             defendingNodes.add(u)
7:         else
8:             attackDefended ← False
9:             break
10:        end if
11:    end for
12:    if attackDefended then
13:        return attackDefended, defendingNodes
14:    end if
15: end for
16: return False, ∅
```

Алгоритам 16 Функција `rouletteDefense`

Улаз: решење s, граф $G = (V, E), k, reducedAttack, defendingNodes, tries > 0$
Излаз: пар *attackDefended*, *defendingNodes*

1: **while** $tries > 0$ **do**
2: $attackDefended \leftarrow$ `True`
3: $tries \leftarrow tries - 1$
4: **for** $v \in reducedAttack$ **do**
5: $candidates \leftarrow []$
6: **for** $u \in N(v)$ **do**
7: $diff \leftarrow getAvailableFAs(u)$
8: $candidates.add(\underbrace{[u, \ldots, u]}_{diff \text{ times}})$
9: **end for**
10: **if** $candidates.size() = 0$ **then**
11: $attackDefended \leftarrow$ `False`
12: **break**
13: **end if**
14: $selectedU \leftarrow candidates.randomElement()$
15: $defendingNodes.add(selectedU)$
16: **end for**
17: **if** $attackDefended$ **then**
18: **return** $attackDefended, defendingNodes$
19: **end if**
20: **end while**
21: **return** `False`, $\emptyset$

Додатак Г

Додатни резултати за k-SRD проблем

Табела Г.1: Упоредни резултати на скупу RANDOM за $n = 10$.

инстанца		k	GREEDY	VNS			ILP		BENDERS		
n	*index*		obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
10	1	2	7	**6**	0.0	0.0	**6**	0.1	**6**	6	0.7
	2		**6**	**6**	0.0	0.0	**6**	0.1	**6**	6	0.7
	3		**6**	**6**	0.0	0.0	**6**	0.1	**6**	6	2.5
	4		6	**5**	0.0	0.0	**5**	0.0	**5**	5	1.2
	5		**6**	**6**	0.0	0.0	**6**	0.1	**6**	6	1.8
10	1	3	8	**7**	0.0	0.0	**7**	0.5	**7**	7	2.4
	2		7	**6**	0.0	0.0	**6**	0.4	**6**	6	3.8
	3		**7**	**7**	0.0	0.0	**7**	0.4	**7**	7	3.0
	4		8	**6**	0.0	0.1	**6**	0.3	**6**	6	2.9
	5		**7**	**7**	0.0	0.0	**7**	0.4	**7**	7	2.9
10	1	4	9	**7**	0.0	0.2	**7**	1.6	**7**	7	3.4
	2		8	**7**	0.0	0.1	**7**	1.6	**7**	7	3.4
	3		8	**7**	0.0	0.7	**7**	1.4	**7**	**7**	4.1
	4		9	**7**	0.0	0.1	**7**	1.3	**7**	**7**	4.2
	5		8	**7**	0.0	0.1	**7**	1.4	**7**	7	4.4
10	1	5	10	**8**	0.0	0.1	**8**	4.1	**8**	8	6.3
	2		9	**8**	0.0	0.0	**8**	2.8	**8**	8	4.4
	3		9	**8**	0.0	0.0	**8**	3.2	**8**	8	11.5
	4		10	**8**	0.0	0.1	**8**	2.7	**8**	8	7.1
	5		9	**8**	0.0	0.0	**8**	2.0	**8**	8	9.7

Табела Г.2: Упоредни резултати на скупу RANDOM за $n = 15$.

инстанца		k	GREEDY	VNS			ILP		BENDERS		
n	*index*		obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
15	1	2	10	**9**	0.0	0.0	**9**	0.9	**9**	9	5.3
	2		**8**	**8**	0.0	0.0	**8**	0.2	**8**	8	2.2
	3		9	**8**	0.0	0.0	**8**	0.2	**8**	8	2.8
	4		10	**9**	0.0	0.0	**9**	1.2	**9**	9	5.0
	5		10	**9**	0.0	0.0	**9**	0.3	**9**	9	3.8
15	1	3	12	**10**	0.0	0.1	**10**	12.2	**10**	10	20.2
	2		**10**	**10**	0.0	0.0	**10**	3.6	**10**	10	12.9
	3		10	**9**	0.0	0.1	**9**	1.5	**9**	9	7.7
	4		12	**10**	0.0	0.1	**10**	12.1	**10**	10	21.3
	5		12	**10**	0.0	0.6	**10**	7.1	**10**	10	22.0
15	1	4	13	**11**	0.0	0.4	**11**	51.7	**11**	11	55.4
	2		12	**11**	0.0	1.1	**11**	37.7	**11**	11	35.4
	3		11	**10**	0.0	2.4	**10**	19.0	**10**	10	36.6
	4		13	**11**	0.0	0.7	**11**	104.8	**11**	11	77.7
	5		14	**11**	0.0	1.3	**11**	80.2	**11**	11	107.0
15	1	5	14	**11**	0.0	4.4	9.3	TL	**11**	11	129.1
	2		13	**11**	0.0	5.9	**11**	239.4	**11**	11	60.0
	3		12	**11**	0.0	1.8	**11**	162.9	**11**	11	72.2
	4		14	**12**	0.0	1.1	12	TL	0	11	TL
	5		15	**12**	0.0	0.9	12	TL	**12**	12	210.0

Табела Г.3: Упоредни резултати на скупу RANDOM за $n = 20$.

инстанца		k	GREEDY	VNS			ILP		BENDERS		
n	*index*		obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
20	1	2	12	**11**	0.0	0.0	**11**	1.5	**11**	11	5.5
	2		13	**12**	0.0	0.1	**12**	0.5	**12**	12	14.0
	3		12	**11**	0.0	0.0	**11**	0.9	**11**	11	6.3
	4		**11**	**11**	0.0	0.0	**11**	0.4	**11**	11	5.8
	5		**12**	**12**	0.0	0.0	**12**	1.4	**12**	12	9.3
20	1	3	14	**12**	0.0	1.3	**12**	7.7	**12**	12	24.9
	2		16	**13**	0.0	0.9	**13**	17.0	**13**	13	109.8
	3		14	**13**	0.0	0.2	**13**	46.2	**13**	13	66.2
	4		14	**13**	0.0	0.7	**13**	14.7	**13**	13	38.0
	5		15	**13**	0.0	0.6	**13**	41.1	**13**	13	62.4
20	1	4	16	**14**	0.0	4.1	**14**	TL	**14**	14	224.2
	2		18	**14**	0.0	17.5	18	TL	0	14	TL
	3		15	**13**	0.0	34.1	23	TL	**13**	13	89.5
	4		16	**14**	0.0	3.2	18	TL	**14**	14	111.2
	5		17	**14**	0.0	46.6	22	TL	0	14	TL
20	1	5	18	**15**	0.0	35.5	-	-	0	14	TL
	2		19	**15**	2.1	51.6	-	-	0	14	TL
	3		16	**14**	0.0	39.0	-	-	0	14	TL
	4		17	**14**	0.0	35.0	-	-	**14**	14	276.2
	5		19	**15**	0.0	36.9	-	-	0	14	TL

Табела Г.4: Упоредни резултати на скупу RANDOM за $n = 45$.

инстанца		k	GREEDY	VNS			ILP		BENDERS		
n	$index$		obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
45	1	2	28	26.7	2.0	16.7	**26**	14.5	**26**	26	240.3
	2		28	**26**	0.0	19.4	**26**	19.5	**26**	26	311.3
	3		28	**26**	0.0	10.5	**26**	35.4	**26**	26	326.0
	4		28	**27**	0.0	13.4	**27**	21.0	0	27	TL
	5		27	**26**	0.0	26.8	**26**	10.3	**26**	26	225.9
45	1	3	35	**29.8**	1.6	178.9	-	-	0	27	TL
	2		34	**29.2**	1.6	197.3	-	-	0	26	TL
	3		35	**30.3**	2.1	186.7	-	-	0	27	TL
	4		35	**29.7**	2.0	206.3	-	-	0	27	TL
	5		34	**29.3**	1.8	213.2	-	-	0	26	TL
45	1	4	40	**32.4**	3.0	130.7	-	-	0	28	TL
	2		39	**32.4**	1.8	213.7	-	-	0	25	TL
	3		40	**32.9**	2.5	293.5	-	-	0	26	TL
	4		40	**32.5**	3.0	260.1	-	-	0	26	TL
	5		38	**32.4**	2.5	110.5	-	-	0	27	TL
	1		43	**35.9**	1.7	241.7	-	-	0	28	TL
45	2	5	42	**36.3**	1.3	234.6	-	-	0	26	TL
	3		43	**37.3**	2.1	354.6	-	-	0	26	TL
	4		43	**36.7**	2.1	303.8	-	-	0	26	TL
	5		42	**35.8**	2.1	323.7	-	-	0	26	TL

Табела Г.5: Упоредни резултати на скупу WIRELESS за $k = 2$.

инстанца		GREEDY	VNS			ILP		BENDERS		
n	R	obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
20	0.3	13	**12.0**	0.0	1.1	**12**	0.25	**12**	12	6.5
	0.4	10	**9.0**	0.0	0.3	**9**	1.16	**9**	9	7.8
	0.5	6	**5.0**	0.0	0.2	**5**	1.14	**5**	5	6.6
	0.6	**3**	**3.0**	0.0	0.0	**3**	0.21	**3**	3	2.6
50	0.3	15	**13.0**	0.0	3.8	-	-	**13**	13	196.2
	0.4	11	**10.0**	0.0	8.3	-	-	**10**	10	177.6
	0.5	8	**6.0**	0.0	4.1	-	-	**6**	6	64.0
	0.6	6	**5.0**	0.0	2.3	-	-	**5**	5	100.1
100	0.3	18	**13.0**	3.0	237.8	-	-	0	13	TL
	0.4	13	**10.0**	0.0	25.8	-	-	0	10	TL
	0.5	**9**	**9.0**	0.0	0.0	-	-	0	7	TL
	0.6	7	**6.0**	0.0	21.5	-	-	-	-	-

Табела Г.6: Упоредни резултати на скупу WIRELESS за $k = 5$.

инстанца		GREEDY	VNS			ILP		BENDERS		
n	R	obj	$\overline{obj}$	$\sigma[\%]$	$\bar{t}_{best}[s]$	obj	$t[s]$	obj	dual	$t[s]$
20	0.3	19	**17.0**	0.0	29.9	-	-	0	16	TL
	0.4	16	**14.3**	3.4	63.5	-	-	0	14	TL
	0.5	9	**8.0**	0.0	7.9	-	-	0	8	TL
	0.6	6	**6.0**	0.0	0.0	-	-	6	6	187.0
50	0.3	**24**	**24.0**	0.0	0.0	-	-	0	16	TL
	0.4	18	**17.8**	0.0	47.7	-	-	0	12	TL
	0.5	14	**13.9**	3.4	43.5	-	-	0	9	TL
	0.6	9	**7.0**	0.0	224.4	-	-	0	7	TL
100	0.3	**33**	**33.0**	0.0	0.0	-	-	-	-	-
	0.4	25	**24.9**	1.8	51.3	-	-	-	-	-
	0.5	**18**	**18.0**	0.0	0.0	-	-	-	-	-
	0.6	12	**11.9**	0.0	139.4	-	-	-	-	-

Биографија аутора

Рођен је 8.1.1997. у Панчеву. Завршио је основну школу и друштвено-језички смер гимназије у Панчеву као носилац Вукове дипломе. Студије информатике на Математичком факултету уписао је 2015. године и на њима дипломирао 2019. са просечном оценом 9,80. Током студија добитник Стипендије за изузетно надарене студенте, као и награде Неделько Парезановић – награде за најбоље студенте рачунарства и информатике на Математичком факултету. Мастер академске студије информатике завршио је 2020. године, одбранивши мастер рад Дигитализација ЕКГ дијаграма. Исте године уписао је докторске студије информатике на Математичком факултету. Све испите на докторским студијама положио је са просечном оценом 10.

Изабран је за сарадника у настави на Катедри за рачунарство и информатику Математичког факултета 2019. године, а у звање асистента 2020. године. Држао је вежбе из курсева Програмирање 1, Програмирање 2, Пројектовање база података, Истраживање података 1, Рачунарска интелигенција и Функционално програмирање на основним студијама, као и Истраживање података на мастер студијама. Научне области интересовања укључују пре свега машинско учење, оптимизацију и биоинформатику. Поред запослења на Математичком факултету, ангажован је и као истраживач у области машинског учења у компанијама *Connect The Dots* (2023–2024) и *BoolSi* (од 2024).

Прилог 1.

Изјава о ауторству

Потписани-а Стефан Капунац

број уписа 2012/2020

Изјављујем

да је докторска дисертација под насловом

Методе за ефикасно решавање доминацијских проблема на великим графовима

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, _____

_____

_

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора _Стефан Капунац_____

Број уписа _____2012/2020_____

Студијски програм ____Информатика_____

Наслов рада _____Методе за ефикасно решавање доминацијских проблема на великим графовима_____

Ментор _____проф. др Александар Картељ_____

Потписани _____

изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, _____

_____

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Методе за ефикасно решавање доминацијских проблема на великим графовима

_____

_____

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, _____

_____