Универзитет у Београду

Електротехнички факултет

Употреба минималних обухватних стабала у апроксимативним алгоритмима за решавање различитих проблема

Мастер рад

Ментор:

проф. др Марко Мишић

Кандидат:

Јован Ђорђевић 3051/2022

Београд, април 2024.

САДРЖАЈ

1. Увод

У данашње време рапидног цивилизацијског напретка, долази се до великих технолошких постигнућа и открића. Самим тим, природно је да се са прикупљањем нових знања о некој научној сфери на том путу откривају и нови проблеми и ограничења. Према томе, и данас постоји велики број проблема из различитих области који још увек немају потпуна, или барем довољно добра и прихватљива решења.

Теорија графова је област математике од велике важности за моделовање великог броја система који се могу срести у стварном животу. Самим тим, природно је да се ова област подробно проучава у контексту информационих технологија, где се, примењујући до сада позната математичка сазнања, конструишу моћни компјутерски алгоритми који имају драгоцену реалну примену.

Минимално обухватно стабло (енгл. *minimum spanning tree* – у даљем тексту и MST) је веома значајна структура података у уској спрези са појмом графа. Овај мастер рад има за циљ да илуструје решавање неких специјалних проблема коришћењем MST. Конкретно, овде ће бити детаљно обрађена два позната проблема: проблем трговачког путника и проблем сегментације слика. Мотивација за овакав избор лежи у великој релевантности ових проблема у областима геолокације и навигације, дизајна сложених структурних шема у електроници, обраде слике, препознавања објеката и др. Ниједан од наведених проблема засад нема прихваћен егзактан оптималан алгоритам, што оставља простора за креативност и иновативност при њиховом решавању. За сваки од предложених проблема биће имплементирана засебна демонстративна апликација.

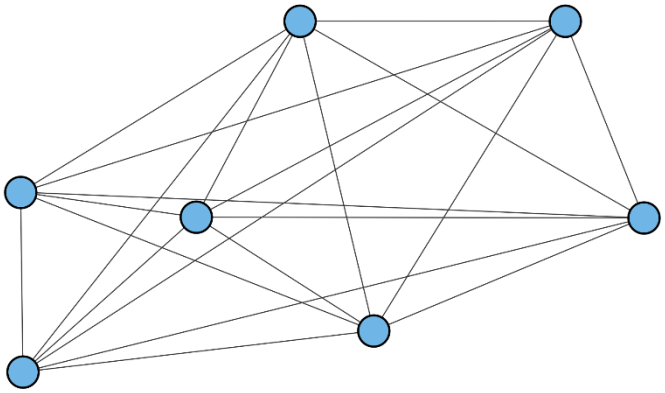
Поглавље 2 резервисано је за неопходан теоријски увод који читаоцу приближава основе теорије графова и минималних обухватних стабала. Биће речи о алгоритмима његовог формирања, као и о неким применама MST у свакодневном животу. Наредна поглавља тичу се одабраних проблема и представљају фокалне тачке овог рада.

У поглављима 3 и 4 биће детаљно обрађени проблем трговачког путника и проблем сегментације слика, респективно. Свако од ових поглавља подразумеваће одговарајући увод у контекст проблема, анализу проблема, предлог и детаљно објашњење адекватног апроксимативног решења проблема (или више њих) уз коришћење MST, навођење одређених имплементационих појединости, и на крају анализу, односно демонстрацију добијених резултата.

Поглавље 5 резервисано је за закључак, где ће се укратко изнети постигнуто у овом раду и продискутовати о могућим побољшањима и даљим токовима развоја предложених решења. У поглављу 6 наводи се коришћена литература за потребе израде овог мастер рада.

2. О МИНИМАЛНИМ ОБУХВАТНИМ СТАБЛИМА

Граф (енгл. *graph*) је нелинеарна структура података која се користи за моделирање и представљање разноврсних релација између делова неког система. Ова структура је погодна због своје флексибилности и применљивости у разним областима, како у научном свету, тако и у свакодневном животу и реалним ситуацијама. Граф G је уређени пар скупова (V, E), где је V коначан непразан скуп, а E представља бинарне релације елемената скупа V. Елементи скупа V називају се чворови, док се елементи скупа E називају гране. Комплетан граф је граф који поседује грану између свака два чвора u и v (Слика 1), док степен чвора u представља број његових суседа. Мултиграф (енгл. *multigraph*) је граф који дозвољава већи број грана између чворова u и v [1].

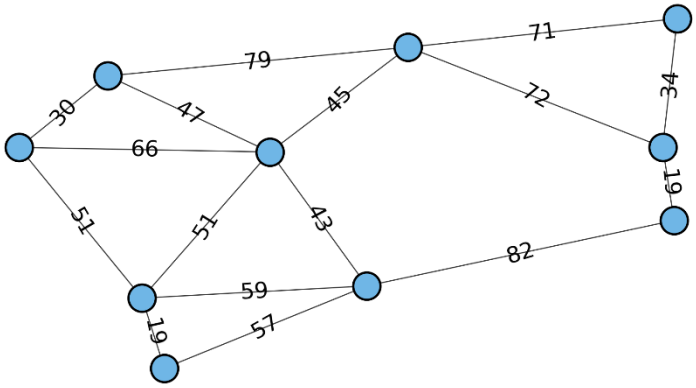


Слика 1. Пример комплетног графа са $|V| = 7$ чворова

2.1. Дефиниција минималног обухватног стабла

Како би се у дискусију увео и појам MST, потребно је да произвољни посматрани граф G има одређене карактеристике, те мора важити да је овај граф [1]:

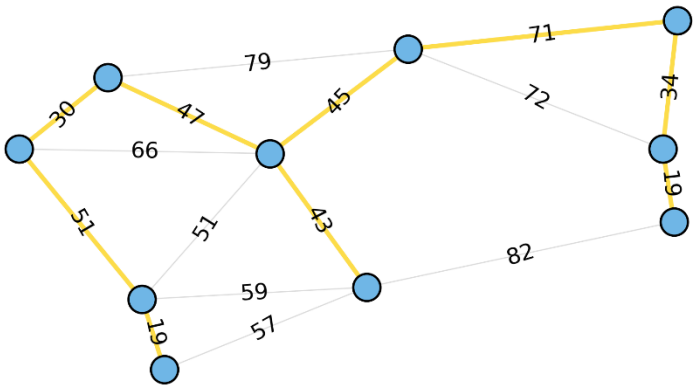
- *повезан* – између свака два чвора u и v у графу постоји пут,
- *неусмерен* – парови чворова (u, v) који одговарају гранама графа су неуређени,
- *тежински* – свакој грани графа је придружена тежина $w(u, v)$.



Слика 2. Пример повезаног неусмереног тежинског графа

Обухватно стабло (енгл. *spanning tree*) повезаног неусмереног графа $G = (V, E)$ представља структуру података моделирану уређеним паром скупова (V', E'), где је $V' = V$, док је $E' \subseteq E$ такав да садржи гране потребне да се повежу сви чворови међусобно, али тако да нема циклуса [1]. Из дефиниције обухватног стабла је јасно да за произвољни посматрани граф у општем случају може постојати више различитих обухватних стабала. Постоје ситуације у којима је потребно одабрати оптимално обухватно стабло, и то оно са најмањом укупном дужином грана. У оваквим случајевима се за моделирање проблема користе тежински графови. Сада можемо дефинисати централни појам овог мастер рада:

Минимално обухватно стабло повезаног неусмереног тежинског графа $G = (V, E)$ је обухватно стабло тог графа које има најмању цену, односно обухватно стабло за које важи да је сума $\sum w(u, v) \mid (u, v) \in E'$ минимална [1].



Слика 3. Гране минималног обухватног стабла (жута боја) за граф са слике 2

Укупна цена минималног обухватног стабла са слике 3 је 359, и може се лако показати да је то најмања укупна цена обиласка свих чворова у датом графу. Наравно, у општем случају може настати ситуација у којој је за одређени граф могуће конструисати више различитих MST, где природа самог проблема диктира одабир одговарајућег стабла.

2.2. Конструкција минималног обухватног стабла

Постоји велики број алгоритама за формирање MST који се извршавају у полиномијалном времену. У наставку ћемо дефинисати и објаснити три одабрана алгоритма: Борувкин алгоритам [2], Крускалов алгоритам [3] и Примов алгоритам [3]. Сви наведени алгоритми припадају категорији похлепних (енгл. *greedy*) алгоритама. Њихови почетни услови наведени су у наставку.

Проблем (Конструкција MST). *Нека је $G = (V, E)$ повезан неусмерен тежински граф са n чворова и m грана. Пронаћи обухватно стабло $MST = (V, E')$ графа G са минималном укупном тежином одабраних грана.*

2.2.1. Борувкин алгоритам

Овај алгоритам настао је 1926. године и представља први откривени алгоритам за конструкцију MST, а послужио је као основа за настанак неких модернијих алгоритама са истим задатком. Претпоставка је да се за примену овог алгоритма користи граф чије су све гране различитих тежина, односно где важи: $e \neq e' \Leftrightarrow w(e) \neq w(e') \mid e, e' \in E$.

Кораци алгоритма:

0. Иницијално, сви чворови графа чине шуму (енгл. *forest*) $F = \{T_1, T_2, \dots, T_n\}$, што представља скуп неповезаних стабала.
1. За свако стабло $T_i \in F$ се пронађе инцидентна грана са минималном тежином.
2. Све одабране гране се укључе у MST, при чему се повезана стабла спајају, а број стабала у шуми смањује.
3. Кораци 1 и 2 се понављају док шума стабала не постане јединствено стабло, и управо оно представља тражено минимално обухватно стабло.

Овај алгоритам користи чињеницу да за сваки чвор у графу важи да ће у оквиру MST бити повезан са остатком графа путем инцидентне гране са најмањом тежином, те ову особину касније проширује и на стабла, која заправо чине повезане компоненте (енгл.

connected components) графа. Како се морају испитати све гране графа, а број стабала шуме се дупло смањује у свакој итерацији алгоритма, коначна временска сложеност Борувкиног алгоритма је $O(m\log(n))$ [2].

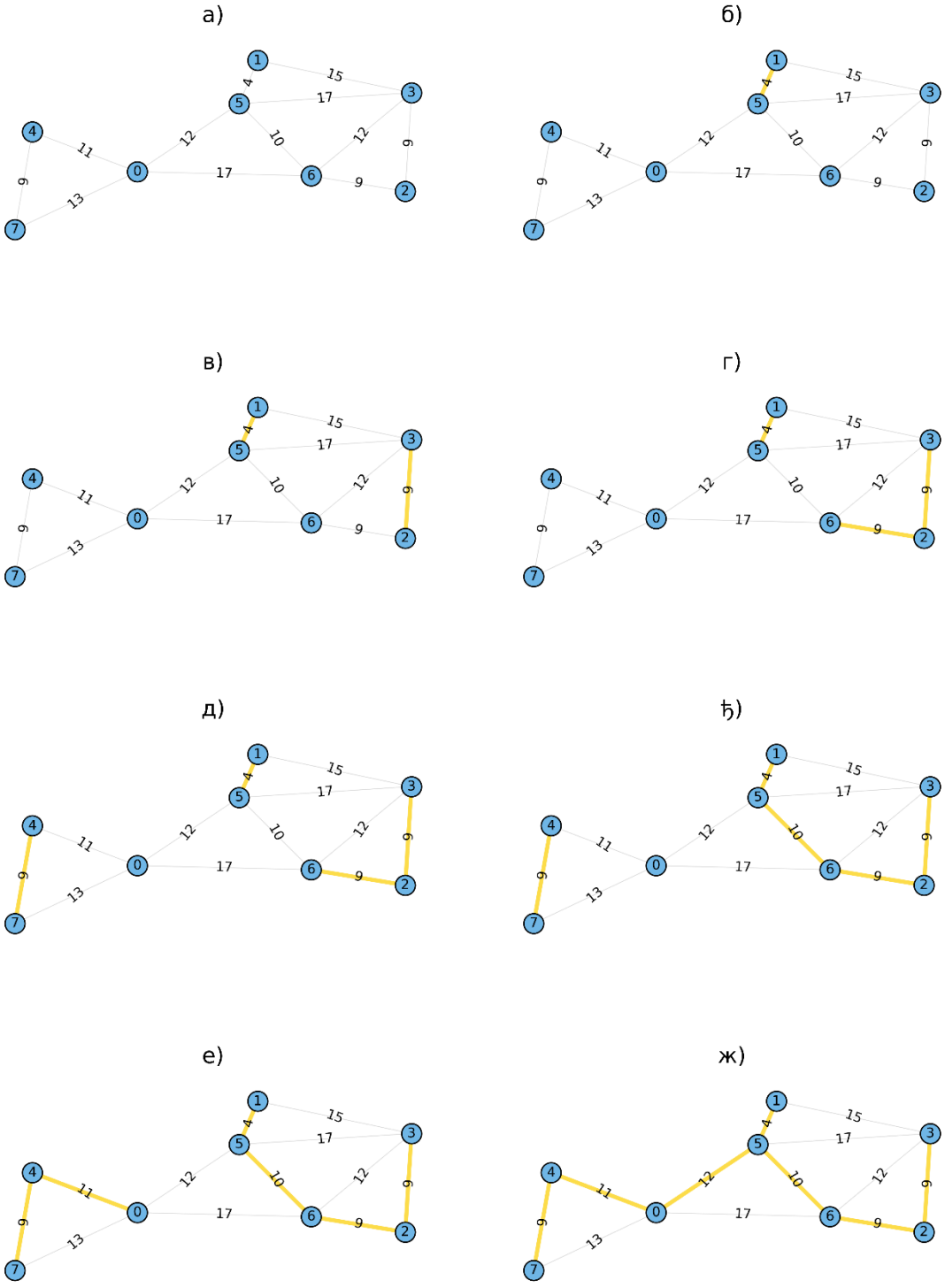
2.2.2. Крускалов алгоритам

Још један типичан похлепан алгоритам за формирање MST је Крускалов алгоритам, настао 1956. године. Он такође користи повезане компоненте, али има одређене разлике у односу на свог претходника.

Кораци алгоритма:

0. Иницијално сви чворови графа чине шуму $F = \{T_1, T_2, \dots, T_n\}$. Све гране се организују у структуру која одржава гране у неопадајућем поретку по тежини.
1. Пронађе се грана графа са минималном тежином чији инцидентни чворови не припадају истој повезаној компоненти (илити грана која не формира циклус).
2. Одабрана грана се уврштава у MST и повезане компоненте које ова грана спаја се обједињују.
3. Кораци 1 и 2 се понављају док број одабраних грана не постане $|V| - 1$, односно док не преостане тачно једна повезана компонента. Све гране ове повезане компоненте формирају тражено минимално обухватно стабло.

Главна разлика између Борувкиног и Крускаловог алгоритма је у томе што Крускалов алгоритам бира гране са минималном тежином на нивоу читавог графа, а не за сваку компоненту засебно. С друге стране, и један и други алгоритам могу се ефикасно имплементирати помоћу структуре дисјунктни-сет (енгл. *disjoint-set*) којом се врше брзе провере повезаности између чворова и сједињавања повезаних компоненти. Сложеност овог алгоритма управо је одређена ефикасношћу поменутих операција и износи $O(m\log(m))$ [1]. На слици 4 илустративно је приказан поступак спровођења Крускаловог алгоритма. Резултујуће минимално обухватно стабло има укупну цену 64 и налази се на одељку слике под ж).



Слика 4. Демонстрација рада Крускаловог алгоритма на графу од $|V| = 8$ чворова

2.2.3. Примов алгоритам

Последњи алгоритам конструкције MST који ће се разматрати у овом раду је Примов алгоритам, осмишљен 1930. године. За свој рад, овај алгоритам захтева одабир произвољног стартног чвора, од којег започиње процес инкременталне градње MST.

Кораци алгоритма:

0. Формира се скуп чворова V' који иницијално садржи произвољно одабран стартни чвор S, као и скуп грана E' који је празан.
1. Пронађе се грана графа са минималном тежином која повезује чворове из скупова V' и $V \setminus V'$.
2. Одабрана грана се додаје у E', а чвор из $V \setminus V'$ се пребацује у V'.
3. Кораци 1 и 2 се понављају док скуп V' не обухвати све чворове графа, односно док не постане тачна једнакост $V = V'$. Све гране из E' чине гране минималног обухватног стабла.

Овај алгоритам истиче да било који чвор графа може да се одабере као корен MST, те зато његова тачност не зависи од одабира почетног чвора. Кључни део у имплементацији овог алгоритма је ефикасно проналажење гране најмање тежине између скупова чворова V' и $V \setminus V'$. Могуће је показати да се временска сложеност Примовог алгоритма може довести до $O(m + n\log(n))$ [1].

2.3. Примене минималних обухватних стабала

Минимална обухватна стабла имају широк спектар примена у различитим сферама науке и индустрије. Пре свега, минимална обухватна стабла доносе велику корист и у свакодневном животу. Примера ради, мотивација за настанак Борувкиног алгоритма била је потреба за дизајнирањем телефонске мреже у Моравији средином 20-их година прошлог века. Задатак је био повезати све градове у тој регији тако да се у том процесу максимално уштеди на телефонским кабловима [2]. У овом смеру се наставило и касније, па се тако данас MST користе за дизајнирање различитих типова важних мрежних система: електронских кола, система за водоснабдевање, компјутерских мрежа, транспортних система, итд.

Ипак, минимална обухватна стабла проналазе своју примену и у комплексним научним и истраживачким областима. Могу послужити као основа за развој многих алгоритама из области теорије графова, а уједно се користе у одређеним оптимизационим дисциплинама попут комбинаторне оптимизације (енгл. *combinatorial optimization*), где се задатак често своди на проналажење конфигурације са оптималним карактеристикама у неком сложеном систему (тематика поглавља 3). Додатно, још једну посебну категорију примена минималних обухватних стабала представља таксономија, што у ширем смислу

представља класификацију датих података на основу одређених карактеристика. Типичне примере таксономије представљају препознавање рукописа и сегментација слика (тематика поглавља 4) [4].

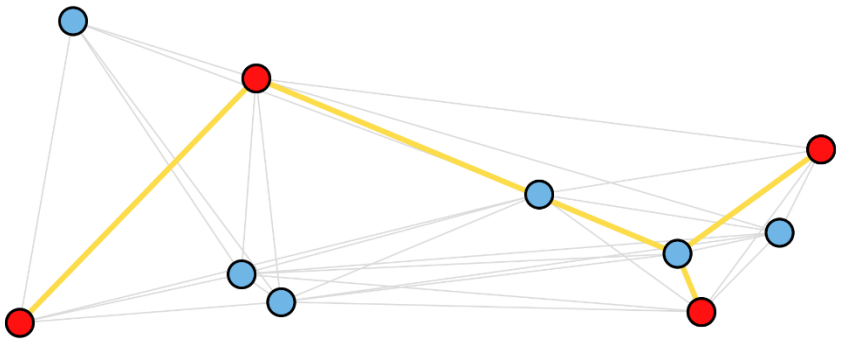
2.3.1. Проблем Штајнеровог стабла

Важно је напоменути још један значајан проблем комбинаторне оптимизације у контексту примене MST, а то је проблем Штајнеровог стабла, који има неколико различитих варијанти. У овом одељку ће кратко бити речи о проблему Штајнеровог стабла у графовима [5].

Проблем (Конструкција Штајнеровог стабла). *Нека је $G = (V, E)$ неусмерен повезан тежински граф са n чворова и m грана, и нека је $T \subseteq V$ скуп посебно одабраних чворова (терминала). Пронаћи обухватно стабло $ST = (V', E')$ које садржи све терминале из T, а може садржати и додатне чворове из V, тако да је укупна тежина одабраних грана минимална.*

Наведени проблем може се посматрати и као генерализација два позната оптимизациона проблема у теорији графова, а то су: проблем проналаска најкраћег пута између два чвора у графу (што је еквивалентно проблему Штајнеровог стабла са тачно 2 терминала) и проблем конструкције MST графа (идентично као проблем Штајнеровог стабла у коме су сви чворови графа терминали). Међутим, иако се оба наведена проблема могу решити у полиномијалном времену, решење у таквом времену не постоји за проблем Штајнеровог стабла у општем случају. Попут минималних обухватних стабала, и Штајнерова стабла имају значајну примену у дизајнирању електронских кола и рачунарских мрежа.

Показано је да наведени проблем спада у категорију НП-тешких проблема (енгл. *NP-hard*), што укратко значи следеће: не само да није решив у полиномијалном времену, већ је на произвољном примеру тешко чак и предложити и доказати оптимално решење [5]. Комплексност овог проблема огледа се у великом броју могућности при одабиру нетерминалних чворова који улазе у састав стабла. Ови чворови називају се и Штајнерове тачке (енгл. *Steiner points*). На слици 5 илустрован је пример проналажења Штајнеровог стабла на произвољном графу са 10 чворова, од којих су 4 чвора терминали означени црвеном бојом. Могуће је уочити да су на датом примеру у састав оптималног стабла укључени и чворови који нису терминали, путем којих се укупна цена стабла успешно минимизовала. Гране Штајнеровог стабла означене су жутом бојом.



Слика 5. Пример проблема Штајнеровог стабла на произвољном графу

Како је до оптималног решења у произвољном случају готово немогуће доћи у коначном времену, овом проблему могуће је приступити на алтернативне начине, што подразумева увођење одређених апроксимација. Управо овде и лежи примена MST, где је користећи наведену структуру или неки од алгоритама формирања MST могуће доћи до одређених задовољавајућих решења проблема Штајнеровог стабла. Услед обимности овог рада, наведени приступи решавању разматраног проблема овде неће бити детаљније обрађени и представљају домен будућих додатних истраживања.

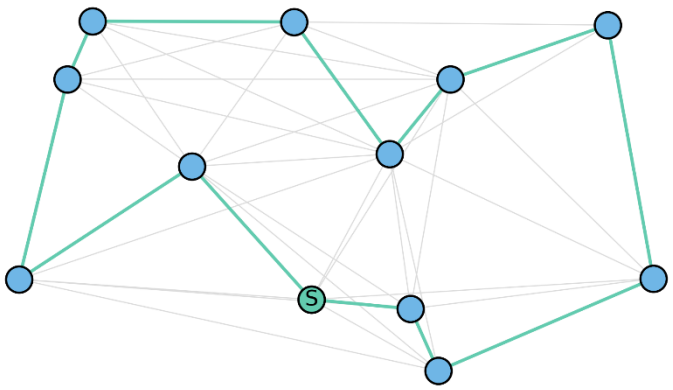
3. Проблем трговачког путника

Проблем трговачког путника (енгл. *Travelling salesman problem* – у даљем тексту TSP) је један од најпопуларнијих оптимизационих проблема који се често среће у многим сферама свакодневног живота, а са великом пажњом се изучава у математичким и инжењерским оквирима. Једна од формулација овог проблема гласи:

„Нека је дат списак градова са тачним удаљеностима између свака два града, и нека је као почетни град произвољно одабран град S. Како одабрати најефикаснију руту обиласка свих градова из S, где се сваки град посети тачно једном, са повратком у S?“ [6]

3.1. Моделовање проблема и почетни услови

Проблем трговачког путника спада у категорију НП-тешких проблема. Први пут је формулисан 1930. године и активно се користи за евалуацију перформанси многих оптимизационих метода [6]. Постоје многобројне познате хеуристике, тако да се за многе практичне примене TSP може или комплетно решити, или апроксимирати са веома великом прецизношћу. На слици 6 је на произвољном примеру илустровано решење TSP, што представља оптималну руту обиласка свих градова са најмањом укупном дужином, са почетком и завршетком у S.



Слика 6. Решење TSP – оптимални обилазак представљен је зеленом бојом

Јасно је да се овај проблем може моделовати тежинским графом $G = (V, E)$, где чворови представљају градове, а гране са својим тежинама повезаност и удаљеност између градова. Овде ће се ради генерализације сматрати да је G комплетан граф, међутим, одређене гране графа које не учествују у добијеним решењима на сликама неће бити приказиване, ради једноставности. Под обиласком овог графа подразумева се обилазак који поштује формулацију проблема TSP и овакав циклични обилазак графа који сваки чвор посећује тачно једном и завршава у полазном чвору назива се Хамилтонов циклус (енгл. *Hamiltonian cycle*) [3]. Овај термин ће бити од додатне важности у наставку овог поглавља. Увешћемо и појам оптималног обиласка OT графа G са почетним чвором у S, који представља обилазак овог графа са минималном ценом (цена обиласка рачуна се као сума тежинâ свих грана које у њему учествују). Постоји неколико познатих варијанти овог проблема, зависно од саме природе његове примене. У овом раду ће од интереса бити метрички *TSP*, што значи да морају бити испуњена следећа својства [7]:

- *Важи правило позитивних тежина*. Удаљеност између чворова u и v је позитивна вредност:

$$u, v \in V \mid w(u, v) > 0. \tag{1}$$

- *Важи правило неједнакости троугла*. Најкраћа удаљеност између чворова u и v је увек директно путем њихове заједничке гране:

$$\forall\, u, v, z \in V \mid w(u, v) \leq w(u, z) + w(v, z). \tag{2}$$

- *Важи правило симетричности*. Удаљеност између чворова u и v иста је као удаљеност између чворова v и u:

$$\forall\, u, v \in V \mid w(u, v) = w(v, u). \tag{3}$$

Сложеност познатих егзактних решења овог проблема је још увек експоненцијална, што је прихватљиво за мањи број чворова, али апсолутно непрактично у неким комплекснијим и реалнијим случајевима. Стога, у таквим ситуацијама се прибегава хеуристикама и апроксимативним алгоритмима који обезбеђују решење које је у одређеној мери приближно оптималном решењу. Међутим, природа овог проблема је таква да на произвољном конкретном примеру оптимално решење није ни познато, нити га је лако пронаћи (НП-тежак проблем). Поставља се питање како у таквом случају евалуирати апроксимирано решење.

3.2. Хеуристичко ограничавање оптималног решења

Пре свега, могуће је извести одређене закључке у погледу оптималног решења, што подразумева грубу процену опсега у коме се цена оптималног решења налази [8, 9]. Управо су у овом процесу од кључног значаја минимална обухватна стабла.

3.2.1. Доња граница оптималног решења

Почевши од анализе доње границе (енгл. *lower bound*) цене оптималног решења, могуће је установити следеће:

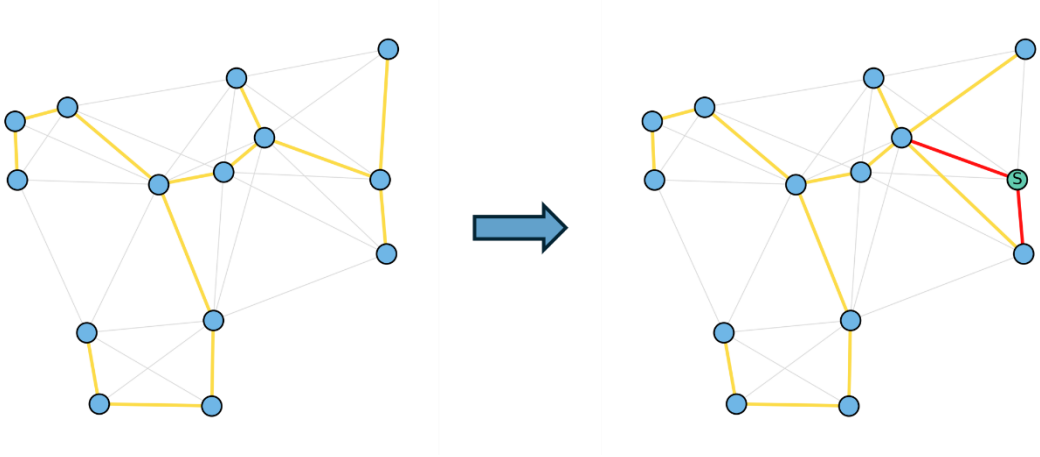
Лема 1. *Цена оптималног обиласка графа G са почетком у чвору S строго је већа од цене минималног обухватног стабла тог графа.*

Доказ: Посматрајмо оптимални обилазак који поседује $|V|$ чворова и $|V|$ грана. Јасно је да он представља циклус, те да се уклањањем било које гране добија обухватно стабло. Како су тежине свих грана позитивне, добијено обухватно стабло засигурно има мању цену од цене оптималног обиласка, док са друге стране по дефиницији мора имати цену већу или једнаку цени минималног обухватног стабла. Према томе:

$$c(OT) > c(ST) \geq c(MST) \;\; \Rightarrow \;\; c(OT) > c(MST), \tag{4}$$

чиме је лема доказана [8].

Могуће је постићи и побољшање у одређивању тражене доње границе, и то помоћу структуре која се формира на начин описан у наставку. Посматрајмо једно минимално обухватно стабло графа G. Одабира се произвољан чвор S и формира се MST без тог чвора, дакле над свим осталим чворовима графа и њиховим међусобним гранама. Потом се бирају две гране са најмањом тежином од чвора S ка његовим суседима, и чвор S се са тим гранама додаје претходно формираном MST. Ова структура назива се 1-стабло (енгл. *1-tree*) [9]. На следећој слици приказан је овај процес, где је прво формирано MST (жута боја), а потом извршена његова трансформација у 1-стабло, односно одабран чвор S са инцидентним гранама (црвена боја).



Слика 7. Процес стварања 1-стабла од минималног обухватног стабла

Евидентно је да је услед додатне гране цена 1-стабла неког графа строго већа од цене MST тог графа, што значи да је доња граница приближнија цени оптималног обиласка. Ипак, потребно је и показати да цена 1-стабла не премашује оптималну цену.

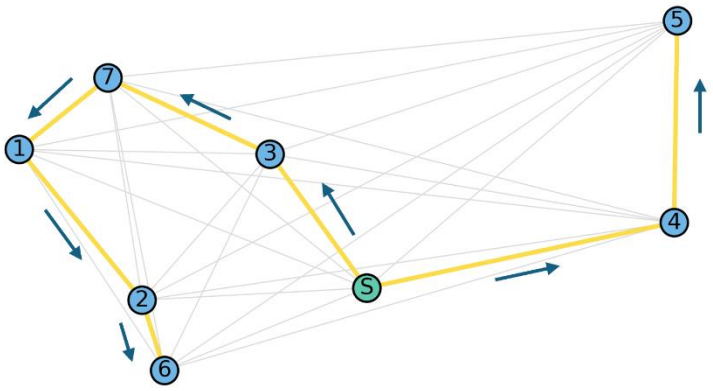
Лема 2. *Цена оптималног обиласка графа G са почетком у чвору S већа је или једнака цени 1-стабла тог графа са одабраним чвором S.*

Доказ: Обе наведене структуре састоје се из две компоненте, а то су: чвор S са две инцидентне гране и преостали чворови са гранама које их међусобно повезују. Како су према дефиницији 1-стабла инцидентне гране чвору S оне са минималном тежином, јасно је да је њихова цена сигурно мања или једнака цени инцидентних грана чвору S у оптималном обиласку. С друге стране, преостали чворови у 1-стаблу су повезани путем MST, што гарантује да је укупна цена грана које их повезују минимална, односно мања или једнака од укупне цене грана које повезују преостале чворове у оптималном обиласку. Комбинујући ова два изнесена закључка, лема је доказана [8].

Сада се доња граница најбоље може одредити тако што се израчуна цена 1-стабла графа за сваки чвор посебно (као одабрани чвор), па се од свих добијених вредности узима највећа, пошто је она најближа оптималној цени. Доња граница заснована на 1-стаблу поприлично је веродостојна, узевши у обзир њену једноставност и лакоћу израчунавања. Постоје неке комплексније технике које обезбеђују већу прецизност, али оне у овом раду неће бити предмет детаљнијег истраживања.

3.2.2. Горња граница оптималног решења

Сада ћемо се позабавити одређивањем горње границе (енгл. *upper bound*) цене оптималног обиласка користећи MST, посматрајући један посебно конструисани обилазак графа, описан у наставку.

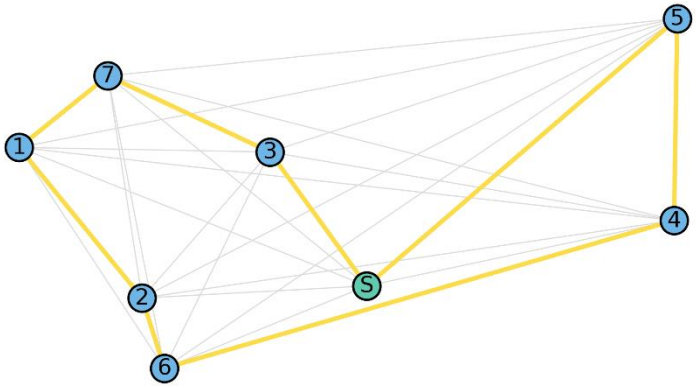


Слика 8. Обилазак минималног обухватног стабла по дубини

Конструише се MST графа G, и потом се обави претрага по дубини (енгл. *depth-first search*) тог стабла, почевши од почетног чвора S (Слика 8). Претрага MST по дубини подразумева обилазак сваке гране тог стабла у оба смера. Нека је M_{dfs} мултиграф који сваки грану из MST садржи по два пута. Тада обилазак MST по дубини представља Ојлеров циклус (енгл. *Eulerian cycle*) E_{cdfs} над M_{dfs}, односно обилазак који сваку грану графа посети тачно једном са повратком у полазни чвор, при чему се води евиденција о посећеним чворовима. На крају обиласка се може анализирати добијена секвенца чворова и од ње конструисати коначна секвенца, и то на следећи начин:

- Ако се чвор није појављивао у коначној секвенци обиласка, уврстити га и означити, пошто је ултимативни задатак TSP посетити све чворове.
- Ако се чвор појављивао у коначној секвенци обиласка, занемарити га, сем ако је у питању последње појављивање чвора S (завршетак обиласка). Овом операцијом се потенцијално уводи попречна грана којом се скраћује пут између чворова. Овај поступак гарантовано може само умањити укупну цену обиласка услед особине неједнакости троугла која важи за метрички TSP, што значи да ће цена директног пута између два чвора бити мања или једнака од цене било ког пута између тих чворова са додатним чворовима.

У примеру са слике 8, након обиласка се добија следећа секвенца чворова, у којој су подвучени редундантни чворови: $S-3-7-1-2-6-\underline{2}-\underline{1}-\underline{7}-\underline{3}-\underline{S}-4-5-\underline{4}-S$. Након уклањања редундантних чворова и додавања попречних грана, коначно се добија жељена путања: $S-3-7-1-2-6-4-5-S$. Илустрација се налази на слици 9.



Слика 9. Коначан приказ обиласка добијеног путем претраге *MST* по дубини

Описани поступак заправо представља формирање Хамилтоновог циклуса H_{cdfs} од Ојлеровог циклуса E_{cdfs} над M_{dfs}, што је веома значајан оптимизациони процес са којим ћемо се и касније сретати. У процесу стварања Хамилтоновог циклуса се из поступка обраде посећених чворова може закључити да у општем случају важи следећа важна неједнакост:

$$c(H_c) \leq c(E_c). \tag{5}$$

Сада можемо донети коначни закључак о горњој граници цене оптималног обиласка.

Лема 3. *Цена оптималног обиласка графа G са почетком у чвору S мања је или једнака двострукој цени минималног обухватног стабла тог графа.*

Доказ: Из претходно наведеног примера је јасно да цена обиласка по дубини мора бити мања или једнака цени двоструког обиласка MST, односно:

$$c\left(H_{cdfs}\right) \leq c\left(E_{cdfs}\right) = 2\,c(MST). \tag{6}$$

Јасно је да сваки предложени TSP обилазак овог графа мора имати цену која је већа или једнака цени оптималног обиласка, што мора важити и за H_{cdfs}, односно:

$$c(OT) \leq c\left(H_{cdfs}\right). \tag{7}$$

Ако укомбинујемо неједнакости (6) и (7), једноставно се показује:

$$c(OT) \leq c\left(H_{cdfs}\right) \leq 2\, c(MST) \;\; \Rightarrow \;\; c(OT) \leq 2\, c(MST), \tag{8}$$

чиме је лема доказана [9].

3.3. Решења проблема заснована на минималним обухватним стаблима

Користећи MST смо успешно одредили и доњу и горњу границу цене оптималног обиласка, што су веома значајни хеуристички подаци за евалуацију апроксимативних алгоритама за решавање TSP. Међутим, суштински најважнији утицај MST при обради овог проблема долази до изражаја када се говори о апроксимацијама оптималног решења, које гарантују довољно добре алтернативе и ограничавају оптимално решење са константним фактором. У наставку ће бити проучене две апроксимације.

3.3.1. 2-апроксимација

Претходни пример није значајан само за одређивање горње границе цене оптималног обиласка, већ и за одређивање 2-апроксимације оптималног решења. Наведени појам значи да се гарантује да ће претходно предложено решење H_{cdfs} имати цену чији је однос са ценом оптималног обиласка мањи од константног фактора 2 [9].

Лема 4: *Цена обиласка H_{cdfs} строго је мања од цене оптималног обиласка графа G са почетком у чвору S помножене константним фактором* 2.

Доказ: Једноставном комбинацијом неједнакости (4) и (6) ,може се показати:

$$c\left(H_{cdfs}\right) \leq 2\, c(MST) < 2\, c(OT) \;\; \Rightarrow \;\; c\left(H_{cdfs}\right) < 2\, c(OT), \tag{9}$$

чиме је тврђење доказано [9].

3.3.2. 1,5-апроксимација (Кристофидисов алгоритам)

Кристофидисов алгоритам (негде и Кристофидис-Сердјуков) је најзначајнији део поглавља које разматра метрички TSP и представља једно од његових најквалитетнијих апроксимативних решења. Овај алгоритам откривен је 1976. године и обезбеђује решење које је у најгорем случају само 50% скупље од оптималног решења, што ћемо касније и доказати [7, 10, 11]. Деценијама се покушавало са унапређивањем овог решења и до данас није било значајнијих помака у томе поступку. Алгоритам је дат у наставку по описним корацима, где ће потом сваки од њих бити детаљније објашњен.

Кораци алгоритма:

1. Формирање MST графа G
2. Одређивање оптималног савршеног упаривања OPM над MST
3. Стварање мултиграфа M_g комбинујући гране MST и OPM
4. Формирање Ојлеровог циклуса E_c над мултиграфом M_g
5. Модификација Ојлеровог циклуса E_c у Хамилтонов циклус H_c

i) Формирање MST графа G

Аутор алгоритма примећује да оптимални обилазак и MST посматраног графа деле значајан број грана, односно да су сличне структуре, те није случајност да је управо конструкција MST полазна тачка овог алгоритма.

ii) Одређивање оптималног савршеног упаривања OPM над MST

Како у оптималном обиласку сваки чвор има степен 2, сличан се ефекат покушава постићи у овом кораку алгоритма. Идеја је допунити MST структуром оптималног савршеног упаривања (енгл. *optimal perfect matching*, у даљем тексту и OPM) која представља подскуп гранâ графа G које повезују чворове са непарним степеном, али тако да збир тежинâ гранâ тог подскупа буде минималан. Међутим, како се савршено упаривање ради за парове чворова, оно је применљиво само ако је број чворова са непарним степеном паран број.

Лема 5. *Број чворова са непарним степеном у графу G је паран број.*

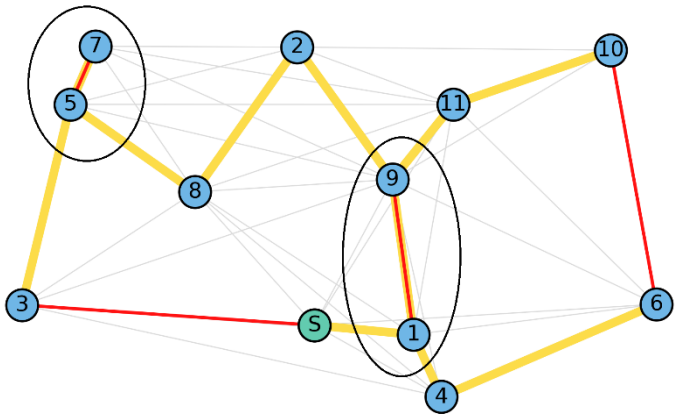
Доказ: По дефиницији, граф G је уређени пар (V, E). Нека су V_{even} и V_{odd} скупови чворова са парним, односно непарним степеном, респективно. Како свака грана утиче на степене два чвора, то значи да је сума степени свих чворова у графу $d_{total} = 2|E|$. С друге стране, укупна сума степени се може израчунати и сабирањем степени свих чворова. Сада важи:

$$|V| = |V_{even}| + |V_{odd}| \;\Rightarrow\; d_{total} = \sum_{i=1}^{|V_{even}|} deg(i) + \sum_{j=1}^{|V_{odd}|} deg(j) = 2|E| \qquad (10)$$

Очигледно је да је сума степени свих чворова d_{total} паран број, а исто важи и за суму степени чворова са парним степеном, како је збир парних бројева паран број. Према томе, и сума степени чворова са непарним степеном мора бити паран број, а пошто је у питању збир

непарних бројева, то значи да је број елемената те суме $|V_{odd}|$ такође паран број, чиме је лема доказана [3].

Број различитих савршених упаривања у комплетном графу са парних n чворова је $(n-1)!!$, и проналажење оптималног савршеног упаривања, илити оног са минималном ценом је засебан оптимизациони проблем који се неће даље разматрати у овом раду. Дакле, у овом кораку алгоритма се идентификују чворови са непарним степеном и конструише се њихово оптимално савршено упаривање OPM. Као угледни пример за демонстрацију рада Кристофидисовог алгоритма користиће се граф са слике 6. Након конструкције MST, уочени су чворови $S, 1, 3, 5, 6, 7, 9, 10$ са непарним степеном. На слици 10 приказани су конструкција MST (жута боја), као и одређивање OPM (црвена боја) за наведене чворове.



Слика 10. Конструкција *MST* и одређивање савршеног упаривања *OPM* са минималном ценом

iii) Стварање мултиграфа M_g комбинујући гране MST и OPM

У општем случају, MST и OPM могу имати и заједничке гране, што се види на претходном примеру (заокружено црном бојом). Стога ће ове две удружене структуре заједно формирати мултиграф M_g.

iv) Формирање Ојлеровог циклуса E_c у мултиграфу M_g

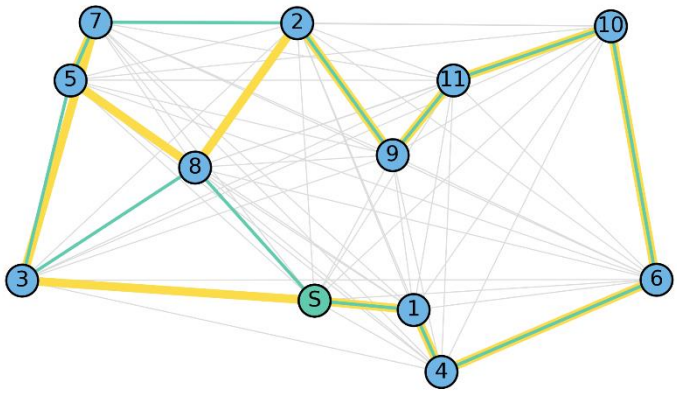
Сада се користи чињеница да сви чворови мултиграфа M_g имају паран степен, што је остварено савршеним упаривањем чворова. Идеја је креирати Ојлеров циклус E_c над M_g који почиње и завршава се у S. Када се овај процес учини на мултиграфу са слике 10, добија се следећи редослед посећивања чворова:

$$S - 1 - 4 - 6 - 10 - 11 - 9 - \underline{1} - \underline{9} - 2 - 8 - 5 - 7 - \underline{5} - 3 - S$$

v) *Модификација Ојлеровог циклуса E_c у Хамилтонов циклус H_c*

Сада се, слично поступку при одређивању 2-апроксимације, из претходно добијене секвенце чворова уклоне поновљене инстанце (подвучено), чиме се уводе попречне гране и добија финална секвенца:

$$\boxed{S-1-4-6-10-11-9-2-8-5-7-3-S}$$



Слика 11. Упоредни приказ: оптимални обилазак (зелена боја), Кристофидисов обилазак (жута боја)

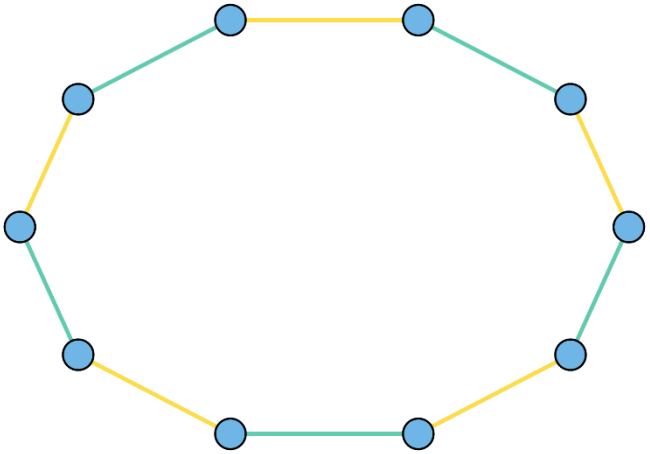
Хамилтонов циклус H_c уједно представља и коначан обилазак, чиме је Кристофидисов алгоритам комплетиран. Цена оптималног обиласка из примера са слике 6 је 489, док је цена обиласка добијеног овим алгоритмом 513, што представља повећање од само ≈4,9%. Добијени резултат је свакако веома задовољавајући, а поготово када се узме у обзир да је, поредећи алгоритме за проналажење оптималне и апроксимиране путање, време извршавања смањено за чак ≈99,9%. Овај податак је драгоцен када се у разматрање узму комплекснији графови са великим бројевима чворова и грана, где није реалистично очекивати да алгоритми за проналажење оптималног обиласка заврше своје извршавање у коначном времену. Ипак, важно је питање може ли се тврдити да ће Кристофидисов алгоритам увек дати довољно добро решење у наведеним границама, са константним фактором од 1,5.

Лема 6. *Цена обиласка H_c добијеног Кристофидисовим алгоритмом спроведеним над графом G са почетком у чвору S строго је мања од цене оптималног обиласка графа G са почетком у чвору S помножене константним фактором 1,5.*

Доказ: Како је мултиграф M_g сачињен од грана̂ MST и OPM, а Ојлеров циклус E_c креиран над M_g, мора важити:

$$c(E_c) = c(MST) + c(OPM). \tag{11}$$

Сада је потребно одредити однос између цене оптималног савршеног упаривања и цене оптималног обиласка. Контрадикцијом је могуће показати да је $c\left(H_{V_{odd}}\right) \leq c(OT)$, где је $c\left(H_{V_{odd}}\right)$ цена оптималног обиласка подграфа графа G сачињеног од чворова са непарним степеном (они улазе у OPM). Нека су PM_1 и PM_2 савршена упаривања чворова из V_{odd} формирана користећи обилазак $H_{V_{odd}}$ одабирајући његове гране наизменично, тако да су њихови скупови грана дисјунктни (Слика 12).



Слика 12. Савршена упаривања PM_1 (жута боја) и PM_2 (зелена боја)

Са претходне слике је јасно да је $c\left(H_{V_{odd}}\right) = c(PM_1) + c(PM_2)$. С друге стране, за OPM над V_{odd} мора важити: $c(OPM) \leq c(PM_1) \wedge c(OPM) \leq c(PM_2)$. Комбинујући неколико претходних неједнакости, добијамо:

$$\begin{aligned} 2\,c(OPM) &\leq c(PM_1) + c(PM_2), \\ 2\,c(OPM) &\leq c\left(H_{V_{odd}}\right) \leq c(OT), \\ c(OPM) &\leq 0{,}5\,c(OT). \end{aligned} \tag{12}$$

Сада се помоћу неједнакости (4), (5), (11) и (12) може установити:

$$\begin{aligned} c(H_c) \leq c(E_c) \;\Rightarrow\; c(H_c) &\leq c(MST) + c(OPM), \\ c(H_c) < c(OT) + 0{,}5\,c(OT) \;\Rightarrow\; c(H_c) &< 1{,}5\,c(OT), \end{aligned} \tag{13}$$

чиме је лема доказана [9].

3.4. Имплементациони детаљи

За потребе испитивања проблема TSP израђена је апликација која имплементира све претходно поменуте алгоритме обиласка графа, врши одређена статистичка израчунавања и обезбеђује одговарајуће илустрације рада ових алгоритама.

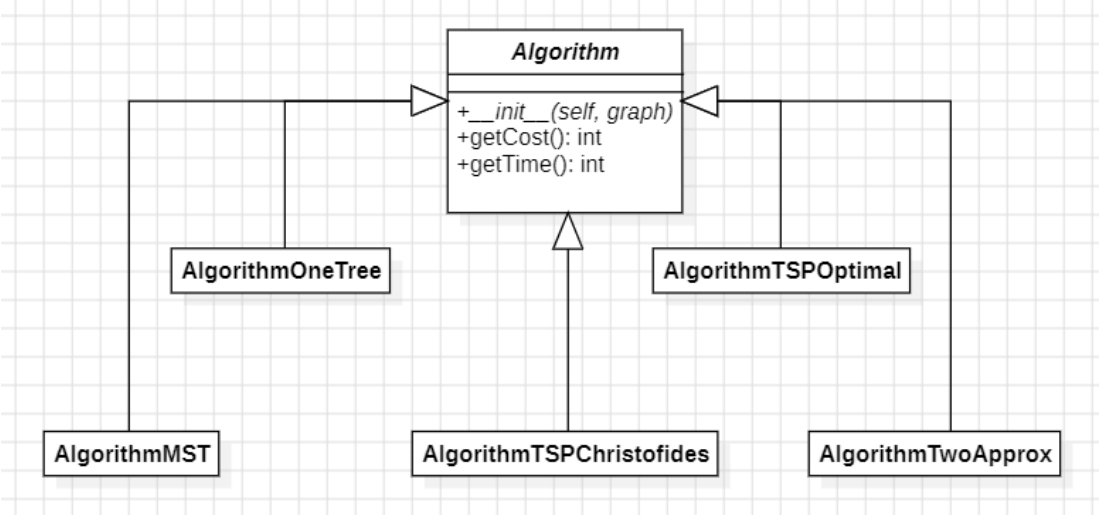
Имплементација апликације одрађена је у програмском језику *Python*, користећи развојно окружење *PyCharm*. За њен развој коришћене су: библиотека *networkx* која служи за рад са графовима, библиотека *matplotlib* употребљена за визуелизацију и библиотека *tsplib95* искоришћена при тестирању. Израда *UML* дијаграма учињена је помоћу алата *StarUML*. Апликација се састоји из три градивна модула, а то су модули *structs, algorithms* и *analysis*.

3.4.1. Модул structs

У оквиру овог модула имплементирана је структура графа коришћена у решавању наведеног проблема. Графови се могу креирати над стварним и над синтетички генерисаним подацима. Графови са стварним подацима формирају се уз помоћ библиотеке *tsplib95* користећи одговарајуће улазне текстуалне фајлове. Графове са синтетичким подацима могуће је створити одабирањем броја чворова и густине повезаности између чворова, док се координате чворова генеришу као случајни бројеви у Еуклидском геометријском простору (енгл. *Euclidean space*), чији је опсег такође могуће подесити. Тежине грана рачунају се као одговарајуће Еуклидске дистанце између чворова. Додатно, овај модул садржи и унапређену имплементацију структуре дисјунктни-сет, која подржава компресију путање и користи се при имплементацији Крускаловог алгоритма (2.2.2).

3.4.2. Модул algorithms

Овај модул се састоји из класа које имплементирају све до сада разматране алгоритме. Све ове класе изведене су из апстрактне класе *Algorithm* која поседује апстрактан конструктор. Наведени алгоритми су имплементирани у складу са детаљном теоријском обрадом сваког од њих у поглављима 2.2.2 и 3. На слици 13 налази се класни дијаграм овог модула, а код 1 представља генералну структуру Кристофидисовог алгоритма.



Слика 13. Дијаграм класа модула *algorithms*

```
class AlgorithmTSPChristofides(Algorithm):
    def __init__(self, g):
        Algorithm.__init__(self, g)

        # Formiranje MST pomocu Kruskalovog algoritma
        mst = AlgorithmMSTKruskal(g).mst

        # Kreiranje optimalnog savrsenog uparivanja OPM
        opm = AlgorithmOPM(g, mst).opm

        # Stvaranje multigrafa Mg od MST i OPM uvrstavanjem svih grana
        mg = nx.MultiGraph()
        mg.add_edges_from(mst.edges())
        mg.add_edges_from(opm.edges())

        # Formiranje Ojlerovog ciklusa Ec nad Mg
        ec = nx.eulerian_path(mg, source=START_NODE)
        ec_node_list = [u for (u, v) in ec]

        # Stvaranje Hamiltonovog ciklusa Hc od Ec
        # Rezultujuca putanja predstavlja finalnu trazenu putanju
        self.path_final, self.cost_final = self.hamiltonian_cycle(ec_node_list)
```

Код 1. Генерална структура Кристофидисовог алгоритма

Како бисмо одредили сложеност Кристофидисовог алгоритма, потребно је анализирати његове кључне елементе. Конструкција MST може се учинити у времену $O(m\log(m))$, најбољи алгоритам проналаска оптималног савршеног упаривања има сложеност $O(n^3)$, док конверзија Ојлеровог у Хамилтонов циклус има линеарну сложеност. Према томе, сложеност Кристофидисовог алгоритма доминантно је обележена одређивањем минималног савршеног упаривања, па овај алгоритам има полиномијалну временску сложеност $O(n^3)$, где је n број чворова (градова). Добијени резултат представља огромно

побољшање у односу на познате алгоритме који гарантују оптималну солуцију и имају експоненцијалну сложеност, од којих је један објашњен у одељку 3.5 [10].

3.4.3. Модул analysis

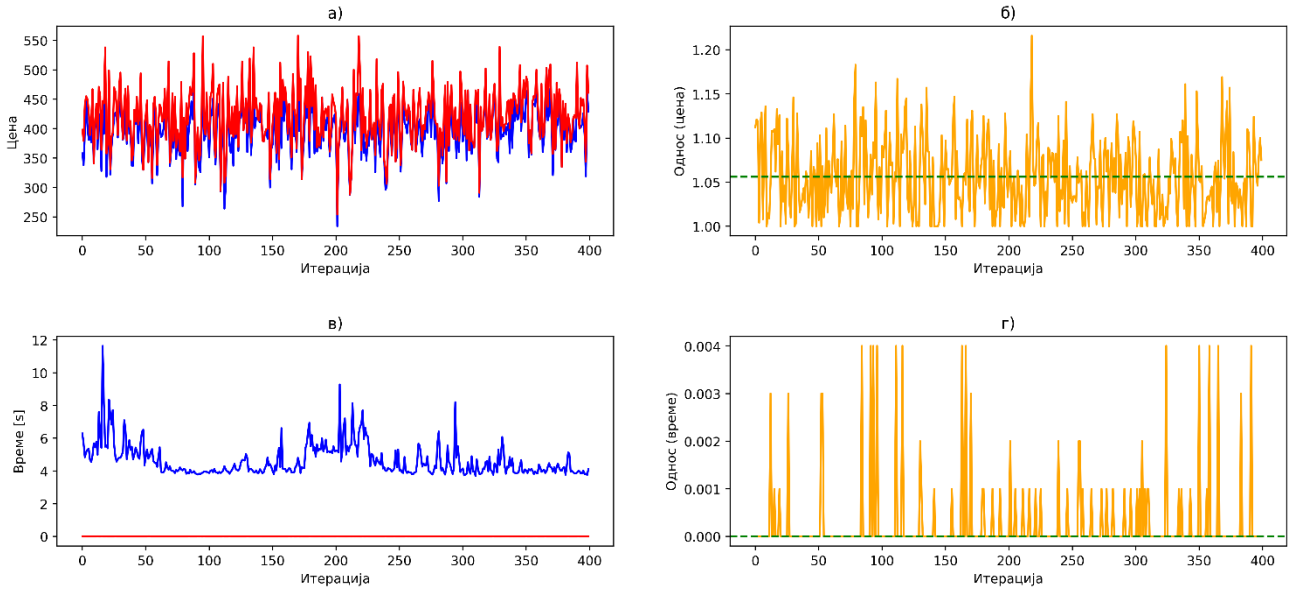
Овај модул има двојаку функцију: може се користити за тестирање и визуелизацију. Тестирањем се могу поредити перформансе различитих алгоритама, и оно се може спровести над стварним подацима (метода *compare_on_real_data*) и синтетичким подацима (метода *compare_on_test_data*). Додатно, овај модул може послужити и за визуелизацију одређеног графа и демонстрацију рада сваког од наведених алгоритама на приложеном графу (метода *visualize_graph_with_layers*). Све илустрације са графовима у овом раду креиране су коришћењем овог модула.

3.5. Резултати

Завршни део поглавља о проблему трговачког путника резервисан је за анализу добијених резултата и адекватне коментаре. Највише пажње посвећено је евалуацији Кристофидисовог алгоритма, као највреднијег постигнућа у оквиру анализе овог проблема, а у контексту примене MST. Сви приказани тестови спроведени су у развојном окружењу *PyCharm* користећи методе модула *analysis* (3.4.3).

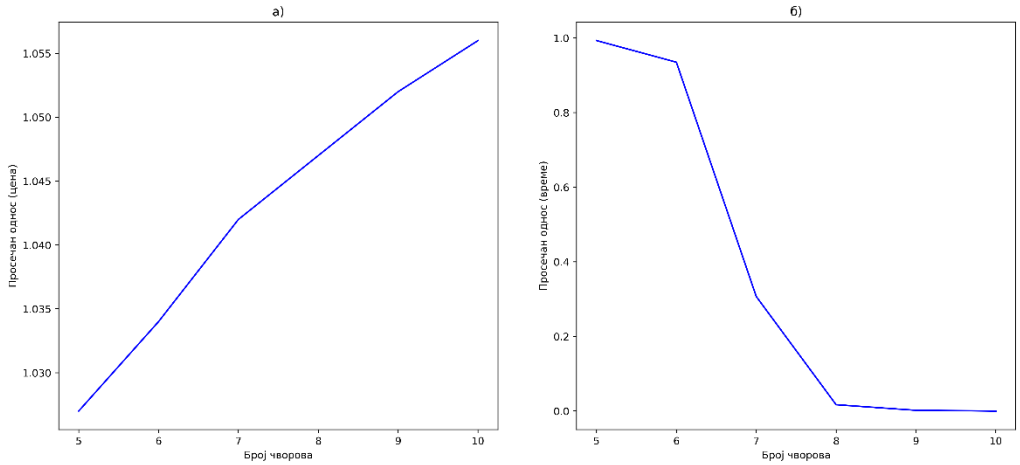
Превасходно је извршено тестирање над синтетичким подацима, услед могућности покретања великог броја различитих тестова и агрегације добијених резултата. Како би се добио реалистичан увид у ефикасност Кристофидисовог алгоритма, било је потребно на неки начин доћи и до оптималног решења, па потом упоредити резултате. Из тог разлога је један од имплементираних алгоритама у модулу *algorithms* (3.4.2) алгоритам примене грубе силе (енгл. *brute force*), који коришћењем обиласка по дубини испробава све могуће комбинације редоследа посећивања чворова и одабира онај поредак обиласка са минималном ценом. Ипак, овај алгоритам има експоненцијалну сложеност, те га је могуће искористити за поређење на графовима са малим бројем чворова.

Путем графика су на слици 14 упоредно приказане перформансе алгоритма примене грубе силе и Кристофидисовог алгоритма на примеру од 400 синтетички генерисаних комплетних графова са по 10 чворова (начин генерисања ових графова описан у одељку 3.4.1). Алгоритми су покренути и упоређени за сваки генерисани граф засебно.



Слика 14. Поређење Кристофидисовог алгоритма (црвена боја) и алгоритма примене грубе силе (плава боја) за сваки од улазних графова, према следећој легенди: а) Приказ добијених цена разматраних алгоритама; б) Однос добијених цена (жута боја); в) Приказ добијених времена извршавања у секундама; г) Однос добијених времена извршавања (жута боја). Просечни односи цена и времена извршавања означени су зеленом испрекиданом бојом на графицима б) и г), респективно

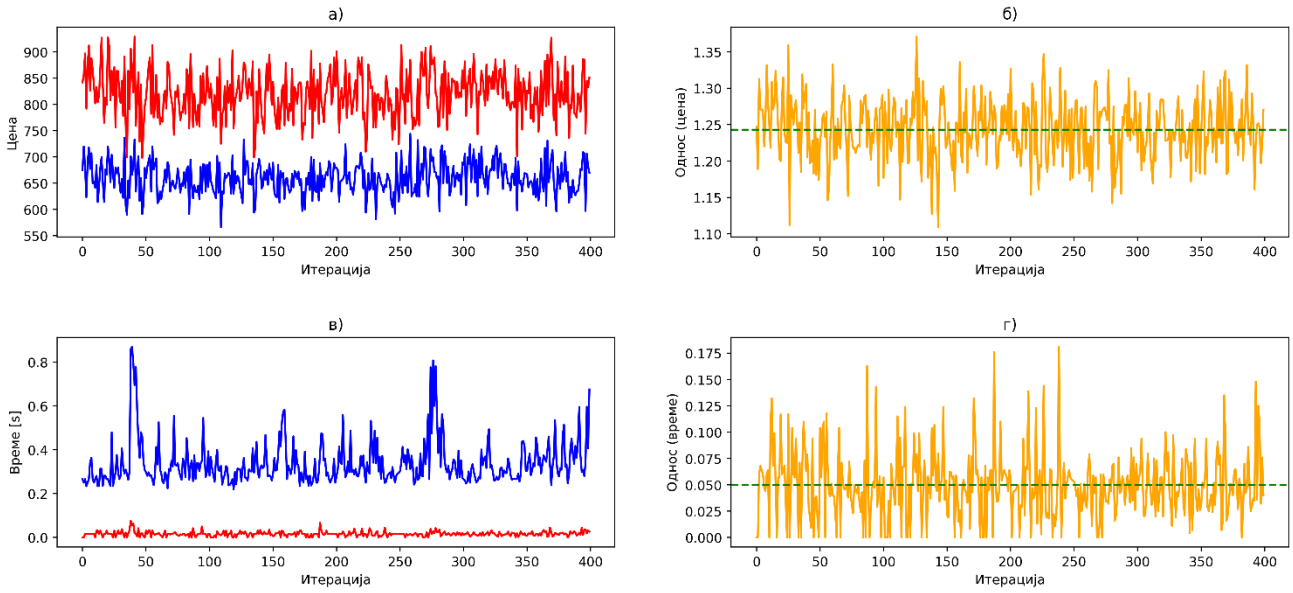
На приложеном узорку Кристофидисов алгоритам даје решење које у просеку увећава цену оптималног решења за само ≈5,6%, што је одличан резултат и показатељ ефикасности примене овог алгоритма у пракси. С друге стране, важно је истаћи да је за очекивати да овај проценат расте са порастом броја чворова, услед драстичног повећања броја путања у графу. Ово је илустровано на слици 15, где је претходни тест репетитивно спроведен над графовима са 5-10 чворова, за сваки број чворова засебно. Из овог разлога је доказ неједнакости (13) који гарантује максимално повећање цене од 50% од пресудног значаја.



Слика 15. Понашање просечног односа а) цене и б) времена извршавања Кристофидисовог алгоритма и алгоритма примене грубе силе са порастом броја чворова

Ипак, учинак из претходних примера још је вреднији ако се узму у обзир и времена извршавања ових алгоритама, где се са слике 14 г) може уочити да уведена 1,5 апроксимација у просеку смањује време извршавања за готово 99,9% у односу на предложени алгоритам за проналажење оптималног обиласка, а са слике 15 је приметно да ће се са порастом броја чворова тај проценат само повећавати, готово до теоретског максимума.

Раније је било речи о ограничавању опсега цене оптималног решења, и тада је установљено да је 1-стабло као процена доње границе веома погодна метрика за евалуацију апроксимативних алгоритама. Ово тврђење илустровано је на наредном примеру, где је на 400 различитих синтетичких графова од по 50 чворова извршено одређивање квалитета решења добијеног Кристофидисовим алгоритмом помоћу 1-стабала над тим графовима (Слика 16).

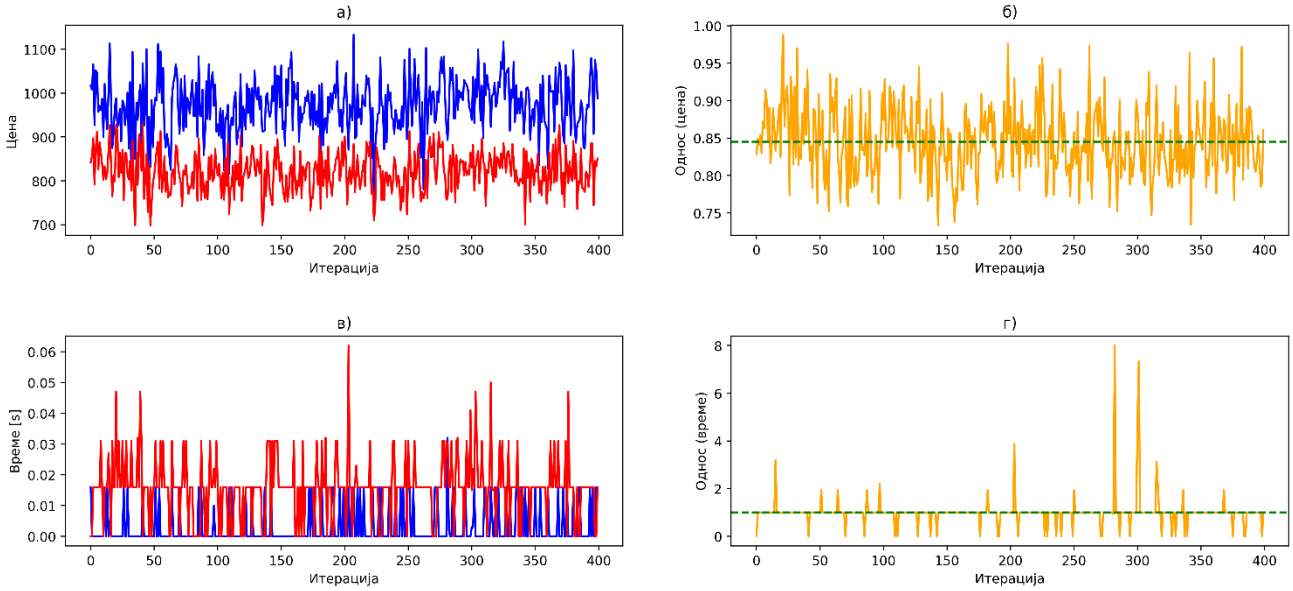


Слика 16. Поређење Кристофидисовог алгоритма (црвена боја) и метрике 1-стабла (плава боја) за сваки од улазних графова, према следећој легенди: а) Приказ добијених цена разматраних алгоритама; б) Однос добијених цена (жута боја); в) Приказ добијених времена извршавања у секундама; г) Однос добијених времена извршавања (жута боја). Просечни односи цена и времена извршавања означени су зеленом испрекиданом бојом на графицима б) и г), респективно

Евидентно је да је услед коришћења апроксимативних алгоритама и хеуристика у овом примеру било могуће значајно увећати број чворова испитиваних графова, чиме се увећава и изазовност проналажења квалитетног решења. У одељку б) може се увидети да је цена решења добијеног помоћу 1,5 апроксимације просечно већа од цене 1-стабла за ≈24,5%, што представља приметан пораст у односу на претходни пример. Но, овај резултат је и за

очекивати, будући да 1-стабло процењује доњу границу оптималног решења, те је у пракси лако установити да оптимални обилазак готово увек има већу цену од ове метрике, па је сходно томе увећан и приказани однос цена. Са друге стране, Кристофидисов алгоритам се поново истиче својом временском ефикасношћу и обезбеђује просечно време извршавања мање од времена извршавања алгоритма формирања 1-стабла за чак ≈95%.

Завршни пример у оквиру тестирања над синтетичким подацима је демонстрација ефикасности Кристофидисовог алгоритма у односу на предложену 2-апроксимацију, и он је илустрован на слици 17. Упоредно извршавање ових алгоритама поново је учињено на 400 различитих синтетичких графова са по 50 чворова. У одељку б) је могуће приметити да Кристофидисов алгоритам у просеку даје решење са ценом мањом за ≈15,5%, што је пожељно и очекивано. Са друге стране, из г) је јасно да Кристофидисов алгоритам постиже просечно време извршавања веће за ≈28.7%. Ипак, када се у обзир узму добијена конкретна времена извршавања, јасно је да је реч о стотим деловима секунде, што је занемарљива разлика у времену извршавања за релативно велики број чворова полазног графа.



Слика 17. Поређење Кристофидисовог алгоритма (црвена боја) и 2-апроксимације (плава боја) за сваки од улазних графова, према следећој легенди: а) Приказ добијених цена разматраних алгоритама; б) Однос добијених цена (жута боја); в) Приказ добијених времена извршавања у секундама; г) Однос добијених времена извршавања (жута боја). Просечни односи цена и времена извршавања означени су зеленом испрекиданом бојом на графицима б) и г), респективно

На крају, понуђена решења проблема трговачког путника тестирана су користећи стварне скупове података. У те сврхе коришћена је библиотека *TSPLIB* [12] која садржи

велики број тест примера у контексту овог проблема, заједно са егзактним оптималним решењима која се користе ради евалуације. У питању су мапе са географским позицијама градова које треба обићи. За потребе овог рада искоришћено је 10 тест примера различитих димензија из наведене библиотеке, док су испитивани Кристофидисов алгоритам и метрика 1-стабла. У табели 1 приказани су резултати рада ових алгоритама на одговарајућим тест примерима.

Табела 1. Приказ резултата рада Кристофидисовог алгоритма и метрике 1-стабла на стварним подацима

Редни број	**Назив теста**	**Оптимално решење (цена)**	**Кристофидисов алгоритам (цена)**	**Кристофидисов алгоритам (време [s])**	**1-стабло (цена)**	**1-стабло (време [s])**
1	a280	2579	2836	1.625	2454	62.384
2	berlin52	7542	8321	0.018	6553	0.394
3	ch150	6528	7072	0.15	5968	9.207
4	eil101	629	697	0.084	564	2.968
5	eil76	538	605	0.049	479	1.385
6	kroA100	21282	23861	0.074	19196	2.485
7	lin105	14379	16432	0.067	13389	3.096
8	pcb442	50778	55527	3.567	46858	270.638
9	pr1002	259045	289791	44.626	225841	4170.284
10	tsp225	3916	4348	0.315	3590	33.879

Димензије коришћених тестова (број чворова графа) могу се закључити из самих назива тестова. Приметно је да се Кристофидисов алгоритам поново показао као веома квалитетна апроксимација оптималног решења, где је апроксимирано решење у просеку скупље од оптималног решења за само ≈11%. Метрика 1-стабла послужила је као солидна апроксимација доње границе оптималног решења, иако је њено време извршавања вишеструко веће у односу на време извршавања Кристофидисовог алгоритма.

4. Проблем сегментације слика

Проблем сегментације слика представља један од најизазовнијих проблема који се могу срести у области компјутерске визије (енгл. *computer vision*). Разумевање садржаја слика и њихово обрађивање у виду препознавања важних целина су процеси без којих је свакодневни живот човека незамислив. Стога, природно је присуство снажне потребе за формализацијом овог проблема и његовим преношењем у рачунарске оквире [13].

4.1. Опис и изазови проблема

Сегментација слика је проблем адекватне поделе слика на одређене визуелне целине (регионе). Још од открића Гешталт принципа у психологији постало је јасно да подела слика на делове и груписање сличних целина представљају основне технике људске визуелне перцепције [14]. Међутим, човек ове процесе извршава несвесно, односно не чини их по унапред дефинисаним критеријумима. Додатно, овде важну улогу игра и искуство, где је људском оку веома једноставно сегментирати неки до тада много пута виђени призор. Примера ради, људско биће може интуитивно разумети слику 18, те на њој са лакоћом може препознати и визуелно контурисати шаховске фигуре. Одавде је јасно да се овом проблему мора пажљиво приступити у рачунарском смислу, где се циљеви морају конкретно формулисати.

Слика 18. Изазов сегментације слике за људско око [15]

Приликом развоја неког програмског метода сегментације слика, од кључног значаја су следеће ставке [16]:

- *Уочавање визуелно хомогених делова слике и њихово прецизно груписање у целине*. Ово је основни задатак сегментације, где највећи изазов представља квалитетно утврђивање граница између суседних региона, односно адекватно одређивање критеријума по којем се врши разграничавање међу регионима.
- *Висока временска ефикасност извршавања*. У циљу практичне примене одређеног метода сегментације слика, од велике је важности да он буде ефикасан. На пример, техника сегментације која завршава свој рад при брзини од неколико фрејмова по секунди (енгл. *frame per second*) могла би послужити у апликацијама за обраду видео снимака.

Чак је и у овој теоретској програмској формулацији јасно да овај проблем није у потпуности егзактан. Како у дефинисању региона и граница између њих постоји доста простора за варијабилност, изгледно је да не постоји универзалан алгоритам који ће давати апсолутно тачно решење за све врсте слика, већ се одабир алгоритма и његових параметара врши на основу конкретне ситуације.

У поглављу 2.3 је као једна од важних примена минималних обухватних стабала наведена таксономија, односно класификовање података на основу одређених особина (у даљем тексту и кластеровање (енгл. *clustering*)). Проблем сегментације слика управо се може посматрати на овај начин, где је пикселе слике потребно груписати у одређене целине (кластере), односно раздвојити их, зависно од њихових карактеристика. Одавде не чуди то што су минимална обухватна стабла вишеструко употребљавана структура у овој области. Постоји неколико различитих приступа овом проблему, али ће се, сходно тематици овог рада, у овом поглављу разматрати графовски приступ сегментацији. Нека је $G = (V, E)$ неусмерен тежински граф са скупом чворова V и скупом грана E. У контексту сегментације слике, чворови из V представљају пикселе слике, а тежине грана из E описују различитост у интензитетима између пиксела. Постоји више различитих начина повезивања ових чворова, а у овом раду разматраће се мрежасти (енгл. *grid*) граф, где су чворови повезани са 8 својих најближих суседа, што се одређује позицијом пиксела на слици. Сегментација S графа G означава поделу скупа V на компоненте (регионе), тако да свака компонента $C \in$

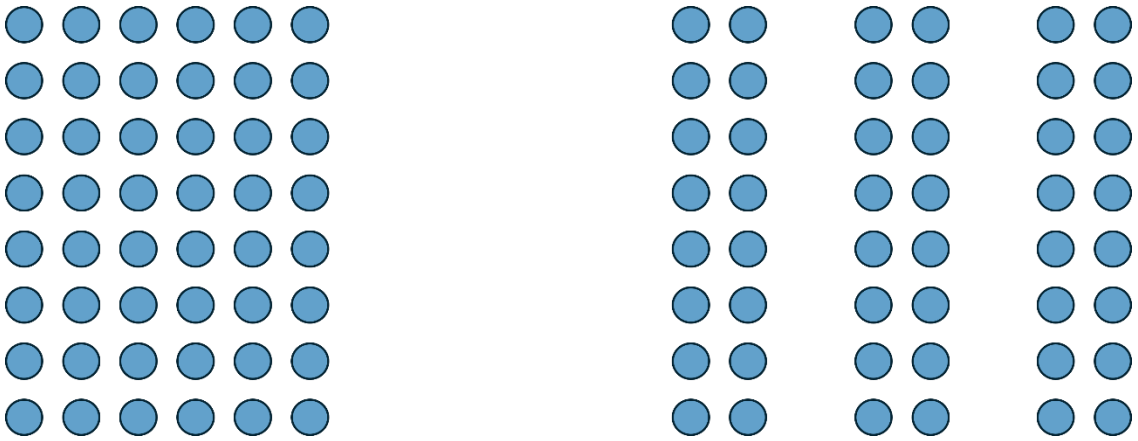
S представља повезану компоненту у графу $G' = (V, E')$, где је $E' \in E$. Другим речима, сегментација S одређена је подскупом скупа грана E. Постоје различити начини мерења квалитета сегментације, али генерална тежња је да елементи исте компоненте буду слични, а елементи различитих компоненти буду међусобно разнолики. Према томе, требало би постићи да гране које повезују елементе исте компоненте имају што мање тежине [16].

4.2. Решења проблема заснована на минималним обухватним стаблима

У овом поглављу биће детаљно описана два алгоритма за сегментацију слика заснована на минималним обухватним стаблима. Прво ће се размотрити један традиционални алгоритам, заснован на сазнањима наговештеним у претходном одељку, а потом ће бити речи о неком модернијем приступу који уводи одређене иновативности.

4.2.1. Занов алгоритам

Први алгоритам који је покушао да искористи карактеристике MST за кластеровање података је Занов алгоритам, настао 1971. године [17]. Аутор своје истраживање заснива на основном Гешталт принципу блискости. На слици 19 је приметно да се у левом одељку све тачке заједнички третирају као део једне целине, док се у десном одељку може извршити интуитивна подела тачака на три одвојена региона.



Слика 19. Демонстрација Гешталт принципа блискости

Подаци које је потребно класификовати приказују се тачкама у дводимензионалном простору, где дистанца између тачака одговара међусобној сличности података. Сличност у оквиру групе података могуће је квантификовати на разне начине, нпр. рачунањем стандардне девијације њихових међусобних дистанци. Међутим, постоје ситуације у којима подаци које је потребно заједнички класификовати нису распоређени равномерном густином

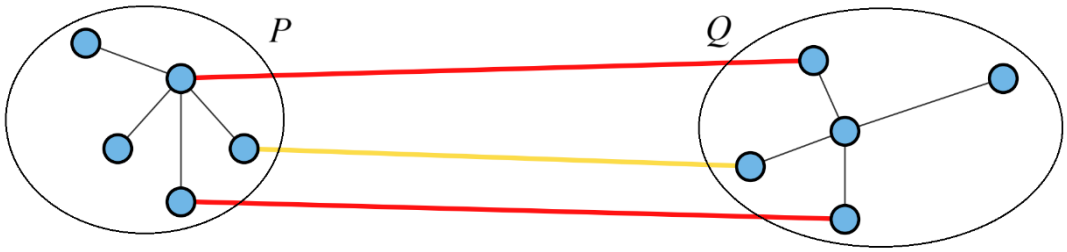
у простору, па у таквим случајевима претходна метрика није довољна. Аутор закључује да је за дати проблем потребно искористити неки локално адаптивни механизам повезивања тачака који предност даје најближим суседима, те је одлука да се у ове сврхе примени минимално обухватно стабло. Ток алгоритма дат је у наставку.

Кораци алгоритма:

0. Проблем се моделује неусмереним тежинским графом G на начин описан у поглављу 4.1.
1. Конструише се минимално обухватно стабло графа G.
2. Идентификују се све неконзистентне гране MST према Зановом критеријуму неконзистентности.
3. Уклоне се све откривене неконзистентне гране, а преостале повезане компоненте MST чине циљне класификационе групе.

i) Мотивација за примену минималних обухватних стабала

Минимална обухватна стабла представљају веома погодну хеуристику у процесу кластеровања због своје организације, где су чворови увек повезани са неколико својих најближих суседа, те MST доприносе у лакшем идентификовању региона, а поред тога су и једноставна за конструкцију. У наставку ће бити речи о неким важним особинама MST у области таксономије. Ипак, да бисмо показали најзначајније теоријске аспекте погодности њихове примене, потребно је увести одређену додатну терминологију. Нека је MST минимално обухватно стабло графа G, дефинисаног у поглављу 4.1. Партиција чворова графа G је подела скупа чворова V на уређени пар дисјунктних подскупова (P, Q). Скуп $set(P, Q)$ је скуп свих грана које повезују P и Q, илити граně чији су инцидентни чворови у P и Q, респективно. Вредност $s(P, Q)$ једнака је најмањој тежини од тежина свих грана које повезују P и Q, а скуп свих таквих грана означен је са $cs(P, Q)$. Из претходних ознака важи $cs(P, Q) \subseteq set(P, Q)$. На слици 20 приказана је партиција (P, Q) произвољног скупа података, где је очигледна циљна подела на кластере P и Q, а где су гране из $cs(P, Q)$ приказане жутом, а преостале гране из $set(P, Q)$ црвеном бојом.



Слика 20. Пример партиције произвољног скупа података

Овде је за очекивати да ће жута грана ући у састав MST, с обзиром на то да она повезује P и Q са минималном ценом, што ћемо у наставку и показати.

Лема 7. *Свако минимално обухватно стабло графа G мора садржати барем једну грану из $cs(P,Q)$, где је (P,Q) произвољна партиција од G.*

Доказ: Ово тврђење једноставно је показати контрадикцијом. Нека постоји MST које не садржи ниједну грану из $cs(P,Q)$. Нека оно садржи грану (u,v) такву да важи $(u,v) \in set(P,Q) \wedge (u,v) \notin cs(P,Q)$. Уочимо сада произвољну грану $(z,t) \in cs(P,Q)$ и укључимо је у MST, тако да се формира нови граф са тачно једним циклусом. Свако обухватно стабло овог графа мора поседовати барем једну грану из $set(P,Q)$, како би P и Q остали повезани. Стога је стабло $MST' = MST \cup \{(z,t)\} \setminus \{(u,v)\}$ такође обухватно стабло графа G и има гарантовано мању укупну тежину од MST, због тога што по дефиницији мора важити $w(z,t) < w(u,v)$. То што значи да полазно MST заправо није минимално обухватно стабло графа G, па је контрадикцијом тврђење доказано.

Сада можемо дефинисати следеће веома важно тврђење које показује суштински значај MST у контексту таксономије.

Лема 8. *Нека је C непразан подскуп чворова од V, нека је $D = V \setminus C$, и нека важи да је $s(P,Q) < s(C,D)$ за све партиције (P,Q) од C. Тада за свако MST графа G важи да његова рестрикција на чворове из C представља повезано подстабло.*

Доказ: Одаберимо произвољну партицију (P,Q) од C. Приметимо да без умањивања општости важи:

$$s(C,D) = s(P \cup Q, D) = min\left(s(P,D), s(Q,D)\right) \Rightarrow s(P,D) \geq s(C,D). \quad (14)$$

Сада је из неједнакости представљене хипотезом и неједнакости (14) јасно:

$$s(P,D) \geq s(C,D) > s(P,Q) \Rightarrow s(P,D) > s(P,Q). \quad (15)$$

Неједнакост (17) указује на чињеницу да ће скуп најкраћих веза $cs(P, S \setminus P)$ бити ограничен скупом $set(P, Q)$, односно да важи $cs(P, S \setminus P) \subseteq set(P, Q)$. Према леми 7, за свако MST графа G важи да оно мора садржати барем једну грану из $cs(P, S \setminus P)$, а самим тим и из скупа $set(P, Q)$, одакле се може закључити да у оквиру посматраног MST скупови P и Q остају повезани за произвољну партицију (P, Q) од C. Према томе, рестрикција MST на C представља јединствену компоненту, односно повезано подстабло, чиме је лема доказана.

Лема 8 испољава својство MST које представља највећу мотивацију за употребу ове структуре у кластеровању. Овим се гарантује да се приликом уклањања одређених грана из MST са великом тежином у циљу поделе на регионе неће нарушити повезаност добијених повезаних компоненти које потенцијално могу представљати резултујуће кластере.

ii) Занов критеријум неконзистентности

Иако се утицај леме 8 показао као значајна полазна тачка за развој овог алгоритма, кључна проблематика се огледа у препознавању грана које нарушавају компактност одређених кластера, односно оних грана чијим би се уклањањем побољшала хомогеност новодобијених кластера. На примеру са слике 20 се могу визуелно јасно уочити два кластера података одређена партицијом (P, Q), где је јасно да је током сегментације потребно уклонити само жуту грану. Поставља се питање како то програмски формулисати.

Нека се посматра произвољна грана e из MST са тежином w и инцидентним чворовима u и v. Аутор усмерава пажњу на гране MST које се налазе у околинама ових чворова. Прецизније, уводи се појам комшилука N_1 чвора u са дубином d, што представља скуп свих грана на путањама дужине највише d које полазе из чвора u и не садрже грану e. Сада се са σ_1 може означити стандардна девијација, а са $\overline{w}_1$ средња вредност тежина грана из N_1. Након што се аналогно дефинишу појмови за чвор v (користећи индекс 2), сада је могуће навести Занов критеријум неконзистентности. Грана e уклања се из MST ако је испуњен барем један од следећа два услова [17]:

$$w > \overline{w}_1 + c * \sigma_1 \wedge w > \overline{w}_2 + c * \sigma_2 \quad (16)$$

$$w > f * max(\overline{w}_1, \overline{w}_2) \quad (17)$$

Параметар c одређује меру у којој се толерише одступање тежине посматране гране од стандардне девијације свих тежина у посматраном комшилуку, док параметар f представља дозвољени однос те тежине и просечне тежине комшилука. Иако на први поглед могу

деловати неинтуитивно, ови критеријуми су потпуно у складу са наведеним циљем локалне адаптивности, где се покушава утврдити да ли посматрана грана e у својој околини представља неки вид статистичке аномалије. Параметар d одређује ниво локалности алгоритма, док параметри c и f одређују толерантност алгоритма на укључивање грана у коначну структуру повезаних компоненти. Веће вредности за c и f подразумевају оштрије критеријуме, а самим тим и мањи финални број повезаних компонената, односно уочених кластера. Резултат алгоритма су повезане компоненте које представљају пронађене кластере података, односно уочене визуелне целине у контексту сегментације слика. На слици 21 приказан је пример рада Зановог алгоритма, где су пронађени кластери обојени произвољним бојама. За демонстрацију је употребљена слика 18.

Слика 21. Пример рада Зановог алгоритма на слици димензија 1920x1080 за узете вредности улазних параметара $d = 3, c = 4, f = 7$

4.2.2. Фелзеншвалбов алгоритам

Фелзеншвалб и Хутенлохер су током 2000-их година приступили сегментацији слика у циљу побољшања до тада познатих класичних алгоритама сегментације, попут Зановог алгоритма. Уочили су неке недостатке у прилагодљивости ових метода на различите типове слика и предложили су одређена побољшања у оквиру самих критеријума одређивања граница између региона. Тако је 2004. године настао Фелзеншвалбов алгоритам [16], детаљно описан у наставку.

Кораци алгоритма:

0. Проблем је моделован неусмереним тежинским графом G на начин описан у поглављу 4.1. Започиње се са сегментацијом у којој сваки чвор графа представља засебну повезану компоненту. Све гране графа се организују у структуру која одржава гране у неопадајућем поретку по тежини.

1. Пронађе се грана графа са минималном тежином и провери се да ли је неопходно ту грану укључити у резултујућу структуру према предложеном критеријуму спајања компоненти K. Ако је то случај, компоненте се повезују.
2. Корак 1 се понавља док се не провере све гране полазног графа. Резултат алгоритма је сегментација S чије компоненте представљају резултујуће регионе слике.

Из наведених корака је јасно да је овај алгоритам урађен по узору на Крускалов алгоритам за конструкцију MST (2.2.2), што наводи на закључак да је резултујућа структура слична MST полазног графа. Међутим, ту не лежи једина мотивација за примену MST у раду Фелзеншвалбовог алгоритма, већ се оно користи и у формулацији критеријума спајања неповезаних компоненти.

i) Фелзеншвалбов критеријум спајања неповезаних компоненти

Сада можемо дефинисати критеријум K који се користи за процену потребе за спајањем две компоненте у сегментацији (два региона слике). Овај критеријум има за циљ повећање локалне адаптивности алгоритма, и то тако што пореди релативне разлике између ивичних елемената суседних региона у односу на интерне разлике између елемената у оквиру тих региона. Како бисмо га прецизно дефинисали, потребно је увести неколико додатних термина [16].

Нека је интерна разлика компоненте $C \subseteq V$ највећа тежина гране која припада минималном обухватном стаблу те компоненте $MST(C, E)$, односно:

$$int(C) = max\ w(e),\ e \in MST(C, E). \tag{18}$$

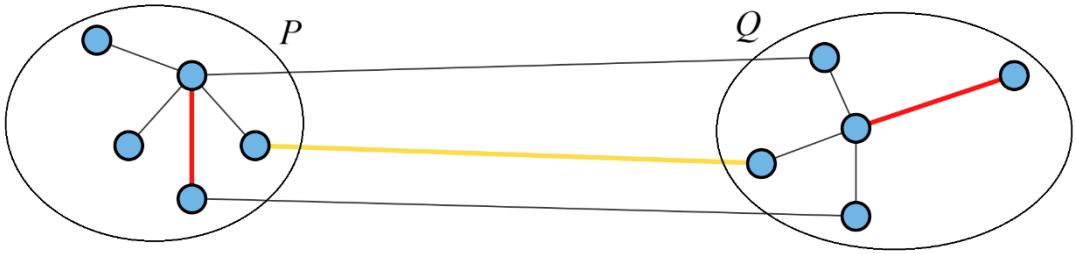
Мотивација за употребу ове вредности лежи у чињеници да компонента C остаје повезана само када се разматрају гране које имају тежину барем $int(C)$, што је важно за идентификацију кластера.

Дефинишимо разлику између две компоненте C_1 и C_2 као вредност једнаку дистанци $s(C_1, C_2)$ формулисаној у поглављу 4.2.1, где су C_1 и C_2 партиције произвољног скупа $C \subseteq V$.

$$diff(C_1, C_2) = s(C_1, C_2) = min\ w(u, v),\ u \in C_1, v \in C_2, (u, v) \in E. \tag{19}$$

Како се током рада алгоритма све гране графа сортирају неопадајуће према тежини, у том поретку ће се разматрати и све гране које повезују компоненте C_1 и C_2. Према томе,

разлика између ових компоненти у кораку 1 увек ће ефективно бити представљена тежином тренутно посматране гране $w(e)$. У пракси може бити проблематично то што у одређивању разлике између две компоненте фигурише само најмања тежина грана између њих, чиме се добија непотпуна слика о начину повезаности ових компонената. Међутим, у пракси се испоставља да овако формулисан критеријум даје задовољавајуће резултате. Могуће је показати да промена овог дела критеријума, таква да се уместо најмање тежине користи медијана или нека друга вредност, резултује алгоритмом сегментације који спада у категорију НП-тешких проблема [18]. Према томе, блага измена критеријума може значајно изменити тежину проблема сегментације. На слици 22 означене су метрике коришћене у критеријуму K, где је као референтни пример послужила партиција (P, Q) са слике 20.



Слика 22. Разлика компонената P и Q (жута боја) и интерне разлике компонената (црвена боја)

Критеријум K процењује да ли има довољно доказа да две посматране компоненте C_1 и C_2 остану раздвојене, односно проверава да ли је разлика између ових компонената $diff(C_1, C_2)$ довољно велика, релативно у односу на интерну разлику барем једне од њих, $int(C_1)$ и $int(C_2)$. Овај однос је условљен и посебном функцијом прага $\tau(C)$, која служи за финије подешавање јачине уведеног критеријума. Ради једноставнијег записа ћемо увести и појам минималне интерне разлике $mint(C_1, C_2)$ која се одређује као:

$$mint(C_1, C_2) = min\ (int(C_1) + \tau(C_1), int(C_2) + \tau(C_2)). \tag{20}$$

Према томе, критеријум K спроведен над компонентама C_1 и C_2 коначно дефинишемо на следећи начин:

$$K(C_1, C_2) = \begin{cases} \text{не спајају се}, & diff(C_1, C_2) > mint(C_1, C_2) \\ \text{спајају се}, & \text{у супротном} \end{cases} \tag{21}$$

Функција прага $\tau(C)$ одређује у којој мери је потребно да разлика између компонената буде већа од њихових интерних разлика, како би то представљало довољно

валидан доказ да је потребно да између њих постоји граница. За мање компоненте, интерна разлика $int(C)$ не представља довољно репрезентативан показатељ локалних карактеристика те компоненте. Стога је природно да функција $\tau(C)$ има већи утицај на критеријум примењен на мање компоненте, те аутор предлаже функцију обрнуто сразмерну величини компоненте:

$$\tau(C) = k \,/\, |C|, \tag{22}$$

где је k произвољан позитиван параметар који се може подешавати. У пракси k одређује толерантност критеријума, те веће вредности параметра k доводе до оштријег критеријума, смањеног броја препознатих граница између компонената, те самим тим и већих компонената. Мање вредности параметра k релаксирају критеријум и повећавају грануларност сегментације [16].

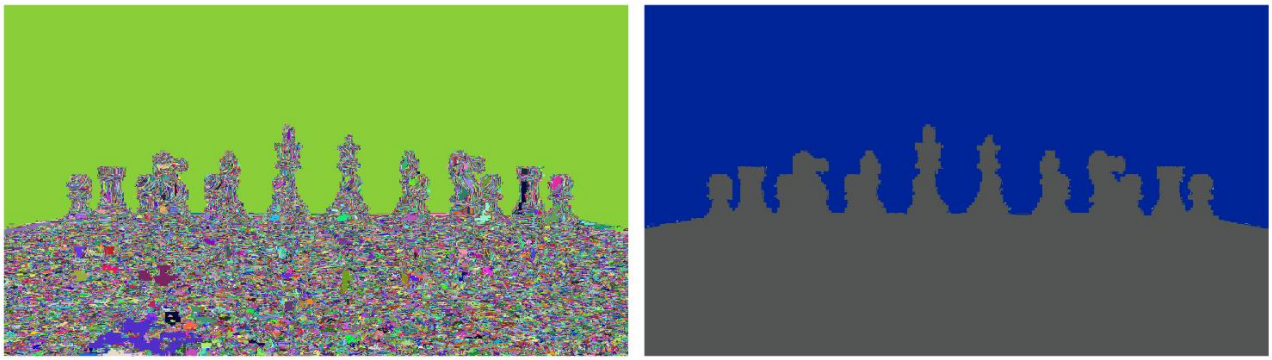
У суштини, $\tau(C)$ може бити било која ненегативна функција и њен одабир неће утицати на валидност самог алгоритма. На пример, могуће је формирати ову функцију тако да критеријум K фаворизује одређене облике препознатих региона слике, тако што се за $\tau(C)$ одређују мале вредности за компоненте непожељних облика (са намером њиховог поновног обрађивања), а велике вредности за компоненте тражених облика (смањена шанса даљег спајања и измене облика).

ii) Прецизност и карактеристике алгоритма

Критеријум K захтева комплексну анализу повезаности чворова, због чега је важно показати да алгоритам даје решење на адекватном нивоу грануларности. Стога, у наставку следи неколико важних дефиниција и лема које утврђују квалитет овог алгоритма [16, 18].

За сегментацију S кажемо да је превише фина ако постоји неки пар компоненти $C_1, C_2 \in S$ за који не постоји доказ о постојању границе између њих према критеријуму K. Овај појам представља сегментацију која има беспотребно велики број компоненти, што у пракси резултује превише сегментираном сликом. Тако можемо дефинисати однос финоће између сегментација S_1 и S_2, где важи да је сегментација S_1 финија од сегментације S_2 ако је свака компонента из S_1 подскуп тачно једне компоненте из S_2, при чему је $S_1 \neq S_2$. На крају, користећи претходно уведене појмове казаћемо да је сегментација S превише груба ако постоји финија сегментација од S која није превише фина. Аналогно дефиницији превише фине сегментације, превише груба сегментација одговара ситуацији у којој слика није довољно сегментирана, те је препознат премали број повезаних компоненти.

На слици 23 дат је упоредни приказ превише фине сегментације S_1 (лево) и превише грубе сегментације S_2 (десно), а демонстрација је одрађена користећи слику 18.



Слика 23. Примери превише фине и превише грубе сегментације

Лема 9. *За сваки граф G постоји сегментација S која није нити превише фина, нити превише груба.*

Доказ: Размотримо сегментацију S_0 графа G у којој сви чворови графа припадају једној компоненти. Очигледно је да ова сегментација није превише фина. Ако S_0 није ни превише груба, лема је доказана. У супротном, по дефиницији превише грубе сегментације закључујемо да постоји сегментација S_1 која је финија од S_0, није превише фина и има већи број компонената од S_0. Овај процес је могуће рекурзивно понављати, чиме се гарантовано у неком тренутку добија не превише груба сегментација, која у граничном случају има максималних $|V|$ компонената. Путем претходног закључка лема је доказана.

Сада ћемо размотрити како Фелзеншвалбов алгоритам дејствује у контексту финоће сегментације:

Лема 10. Сегментација S *добијена Фелзеншвалбовим алгоритмом користећи критеријум K није превише фина.*

Доказ: Ово тврђење могуће је показати контрадикцијом. Претпоставимо да је сегментација S превише фина, што по дефиницији значи да постоји неки пар компоненти за који критеријум K није испуњен. Дакле, мора постојати барем једна грана која повезује такав пар компоненти, а која је разматрана у кораку 1 овог алгоритма и није проузроковала спајање. Нека се посматра прва таква грана $e_q = (u_i, u_j)$ на коју се наишло у итерацији q овог алгоритма и нека су у питању компоненте C_i^{q-1} и C_j^{q-1}. Како ова грана није проузроковала

спајање, важи да је $w(e_q) > mint(C_i^{q-1}, C_j^{q-1})$. С друге стране, приметно је да, у случају да у неком кораку алгоритма не дође до спајања компонената C_i и C_j, једна од ових компонената завршава у свом облику у финалној сегментацији. Ово је уочљиво услед обраде грана по неопадајућем поретку по тежини. Према томе, мора важити $C_i = C_i^{q-1}$ или $C_j = C_j^{q-1}$. У сваком случају, очигледно је да је $w(e) > mint(C_i, C_j)$, односно да је критеријум K испуњен за C_i и C_j. Ово је контрадикција у односу на полазну претпоставку, чиме је лема доказана.

Лема 11. Сегментација S *добијена Фелзеншвалбовим алгоритмом користећи критеријум* K *није превише груба.*

Доказ: Ову лему такође можемо доказати контрадикцијом. Претпоставимо да је сегментација S превише груба, што значи да постоји финија сегментација T од S која није превише фина. Закључујемо да мора постојати барем једна грана e интерна за компоненту $C \in S$, а која повезује две компоненте $A, B \in T$, такве да је $A \subset C$ и $B \subset C$. Како је у оквиру сегментације T критеријум K испуњен за A и B, важи $w(e) > mint(A, B)$. Нека је без умањивања општости $w(e) > int(A) + \tau(A)$. Услед ове неједнакости и редоследа обраде грана, алгоритам је прво морао размотрити све гране у $MST(A, E)$, па тек онда гране које повезују A са другим подкомпонентама од C. Одавде је јасно да је у оквиру сегментације S компонента A морала бити формирана пре C, тако да је приликом стварања C у неком тренутку дошло до спајања A и неке друге компоненте. Тежина гране која је изазвала спајање морала је бити барем $w(e)$, међутим то би по дефиницији критеријума K значило да је $w(e) \leq int(A) + \tau(A)$, што је контрадикција, па је овим лема доказана.

Доказивањем лема 10 и 11 потврђено је да Фелзеншвалбов алгоритам врши квалитетну процену неопходне грануларности решења, у зависности од уведеног параметра k. На слици 24 приказан је пример рада Фелзеншвалбовог алгоритма, где су пронађени кластери обојени произвољним бојама. За ову демонстрацију искоришћена је слика 18.

Слика 24. Пример рада Фелзеншвалбовог алгоритма на слици димензија 1920x1080 за узету вредност улазног параметра $k = 100$

4.3. Имплементациони детаљи

За потребе испитивања проблема сегментације слика израђена је апликација која имплементира оба претходно поменута алгоритма, тестира их и упоређује на предодређеном скупу података и обезбеђује одговарајуће илустрације.

Имплементација апликације одрађена је у програмском језику *Python*, користећи развојно окружење *PyCharm*. За њен развој коришћене су следеће библиотеке: библиотека *networkx* која служи за рад са графовима, библиотека *matplotlib* употребљена за визуелизацију и библиотека *numpy* која је послужила за одређена математичка израчунавања. Израда *UML* дијаграма учињена је помоћу алата *StarUML*. Апликација се састоји из три основна модула, а то су *structs*, *algorithms* и *analysis*.

4.3.1. Модул structs

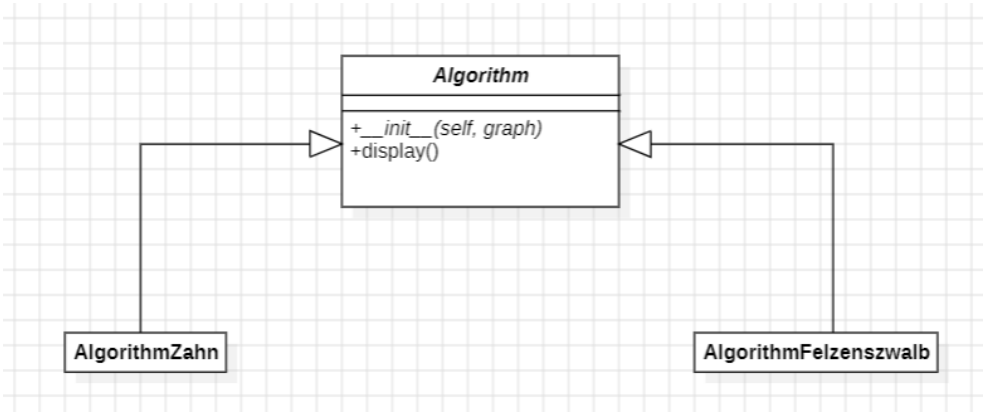
У оквиру овог модула имплементирана је структура графа коришћена у решавању наведеног проблема. Граф се на основу задате слике за сегментацију формира на начин описан у одељку 4.1. Слика се представи као низ пиксела који се потом секвенцијално обрађује, израчунавају се тежине грана и на крају формира циљни мрежасти граф. Ипак, овде је важно разликовати случајеве монохроматских (једнобојних) слика и слика у боји. У случају да се сегментира једнобојна слика, пиксели се описују векторима дужине 1 са могућим вредностима 0-255. Тежине грана се рачунају као $w(v_i, v_j) = |i(v_i) - i(v_j)|$, где пикселима p_i и p_j одговарају чворови v_i и v_j, као и интензитети $i(v_i)$ и $i(v_j)$, респективно. Са друге стране, слике у боји садрже пикселе са описним векторима дужине 3 (нпр. систем RGB вредности), такође са опсегом 0-255. У том случају, тежине грана представљене су

торкама (енгл. *tuple*) чије се вредности рачунају аналогно претходном случају, за сваки индекс засебно. При обради слика у боји, оба алгоритма извршавају се по 3 пута (по једном за сваку од RGB вредности), и оба узимају опрезнији приступ формирања компоненти. Занов алгоритам задржава грану у финалној сегментацији само у случају да се она покаже као конзистентна у сва 3 случаја, док Фелзеншвалбов алгоритам два пиксела ставља у заједничку компоненту само у случају да су они сврстани у исту компоненту у сва 3 покретања.

Додатно, овај модул садржи и унапређену имплементацију структуре дисјунктни-сет, која подржава компресију путање и користи се при имплементацији Фелзеншвалбовог алгоритма.

4.3.2. Модул algorithms

Овај модул се састоји из класа које имплементирају претходно разматране алгоритме. Обе класе изведене су из апстрактне класе *Algorithm* која поседује апстрактан конструктор. Имплементација ових алгоритама потпуно је у складу са њиховом детаљном теоријском анализом представљеном у одељку 4.2.



Слика 25. Дијаграм класа модула *algorithms*

Додатно, у имплементацији Зановог алгоритма се конструкција полазног MST врши помоћу Крускаловог алгоритма (2.2.2), мада је могуће користити и неки други приступ. Претрага комшилука посматраног чвора и прикупљање одговарајућих тежина грана врши се рекурзивним обиласком по дубини. Гране MST не обрађују се по неком утврђеном поретку, што не утиче на валидност алгоритма.

У Фелзеншвалбовом алгоритму се у кораку 0 гране сортирају по неопадајућем поретку по тежини користећи неки од познатих алгоритама сортирања. Сегментација се одржава помоћу структуре дисјунктни-сет, која обезбеђује ефикасне операције провере припадности

два чвора истој компоненти и спајања компонената. Током обраде критеријума спајања неповезаних компоненти Фелзеншвалбовог алгоритма је речено да ће се, услед обраде грана у утврђеном редоследу, разлика компонената C_i и C_j ефективно одређивати као тежина тренутно посматране гране $e = (v_i, v_j)$. Додатно, из истог разлога је могуће закључити да ће, у случају спајања ових компоненти у C_{ij}, вредност $int(C_{ij})$ гарантовано бити једнака $w(e)$, односно да ће највећа тежина гране у MST новоформиране компоненте увек бити једнака тежини гране која је учествовала у формирању те компоненте. Величина компоненте C_{ij} се једноставно одређује као збир величина компонената C_i и C_j. У наставку је дат код 2 који представља генералну структуру Фелзеншвалбовог алгоритма.

```
class AlgorithmFelzenszwalb(Algorithm):
    def __init__(self, g, k):
        Algorithm.__init__(self, g)
        self.k = k

        # Formiranje pocetne segmentacije S od svih cvorova i bez grana
        self.s = nx.Graph()
        self.s.add_nodes_from(self.graph)

        # Sortiranje grana grafa neopadajuce po tezini
        edges = sorted(self.graph.edges(data='weight'), key=lambda x: x[2])

        # Stvaranje strukture disjunktni-set
        dsuf = DisjointSetUnionFind(self.graph.number_of_nodes(), k=self.k)

        # Obilazak svih grana grafa
        for u, v, weight in edges:
            c1 = dsuf.find(u)
            c2 = dsuf.find(v)

            # Provera povezanosti komponenata
            if c1 == c2:
                continue

            # Ako kriterijum K nije ispunjen, komponente se spajaju
            if weight <= dsuf.mint(c1, c2):
                dsuf.union(c1, c2, weight)
                self.s.add_edge(u, v, weight=weight)

        # Prikaz rezultujucih povezanih komponenti segmentacije S
        self.cc = nx.connected_components(self.s)
        self.finalize()
```

Код 2. Генерална структура Фелзеншвалбовог алгоритма

Сада се можемо позабавити временском сложеношћу предложених алгоритама. Како се оба алгоритма спроводе над мрежастим графом, за такав се граф може уочити линеарна зависност $m = O(n)$, где је m број грана, а n број чворова. Занов алгоритам се састоји од

конструкције MST и уклањања неконзистентних грана. Конструкција MST резултује сложеношћу $O(m\,log(m))$, док се обилазак околине чворова по дубини врши за сваку грану MST (које по дефиницији има $n-1$ грана) и подразумева истраживање комшилука до нивоа d, што представља укупну сложеност $O(n\,deg^d)$, где је deg степен посматраног чвора. Како се у пракси за d узимају мале вредности, и степен чвора у MST је такође генерално мали (најчешће 2), могуће је закључити да је сложеност Зановог алгоритма доминантно одређена управо конструкцијом MST, те она износи $O(m\,log(m))$, што се на крају своди на $O(n\,log(n))$ за n пиксела. С друге стране, Фелзеншвалбов алгоритам сачињен је од сортирања свих грана, а потом и спровођења критеријума над сваком од њих. Сортирање грана подразумева сложеност $O(m\,log(m))$, док се при спровођењу задатог критеријума врше одређене операције над структуром дисјунктни-сет, те је укупна сложеност овог дела алгоритма $O\big(m\alpha(m)\big)$, где је $\alpha(m)$ веома споро растућа инверзна Акерманова функција [3]. Генерална сложеност Фелзеншвалбовог алгоритма одређена је управо кораком сортирања, те се одређује као $O(m\,log(m))$, што се на крају своди на $O(n\,log(n))$ за n пиксела. Овај закључак је у складу са уоченом сличношћу овог алгоритма са Крускаловим алгоритмом, који је окарактерисан идентичном временском сложеношћу.

У пракси, слике које се користе као улазни подаци у овим алгоритмима потребно је претходно обрадити, тако што се оне провлаче кроз Гаусов филтер за уклањање шума, где је експериментално одабрана Гаусова константа $\sigma = 0.8$. Узета вредност не чини никакве видљиве промене на сликама, а побољшава дејство обрађених алгоритама [16].

4.3.3. Модул analysis

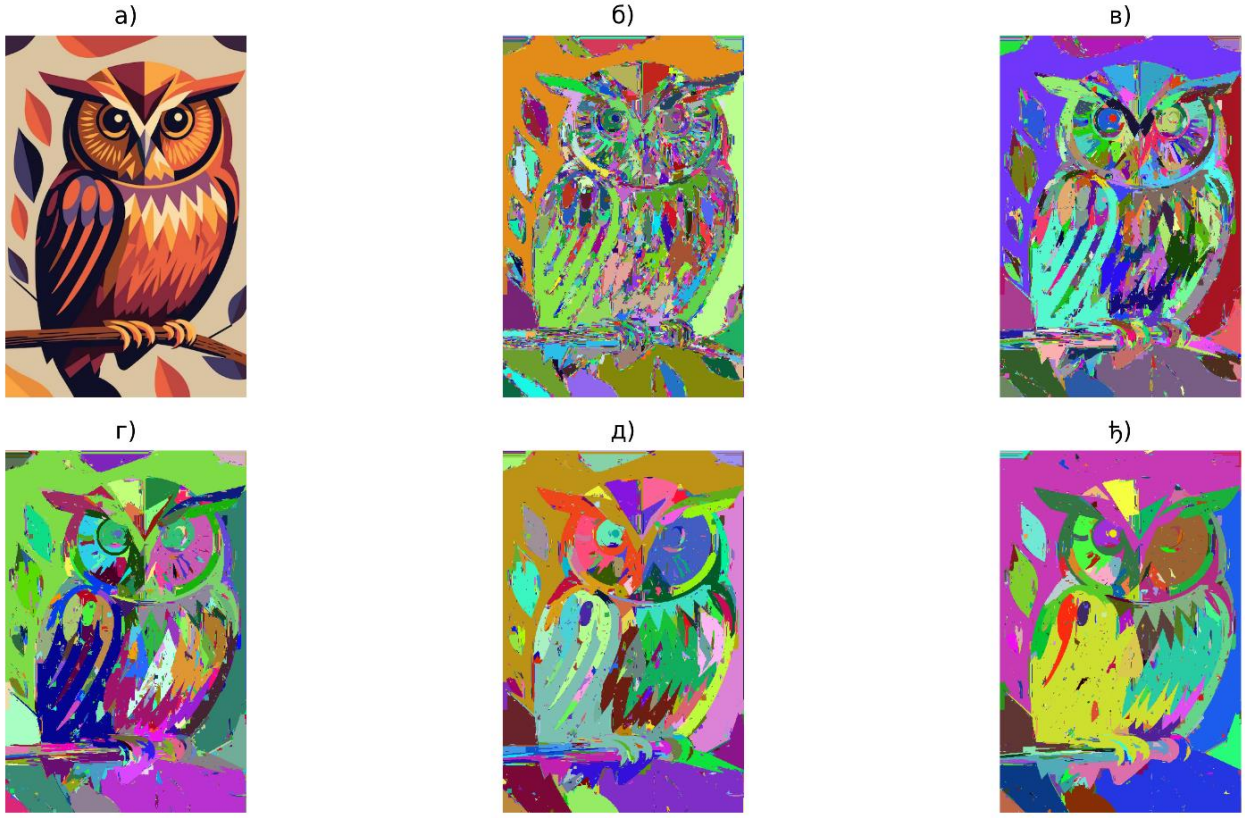
Овај модул служи за спровођење тестова сегментације слика користећи оба обрађена алгоритма. Помоћу методе *segment_image* је могуће одабрати слику и специфицирати опсеге свих респективних параметара (d, f и c за Занов алгоритам, односно k за Фелзеншвалбов алгоритам), те добити шематски приказ сегментација чија се грануларност градацијски мења. Метода *comparison_test* обезбеђује упоредни приказ рада ових алгоритама уз одабрану слику и задате параметре алгоритама.

4.4. Резултати

Ово поглавље резервисано је за анализу резултата рада представљених алгоритама за сегментацију слика за различите врсте слика и вредности параметара. Иако се евалуација

њиховог квалитета у контексту времена извршавања може прецизно учинити, исто не важи и за процену квалитета добијених сегментација, већ се то чини одока и субјективно. За потребе спровођења алгоритама сегментације слика коришћене су, како синтетичке слике, тако и природне слике из стварног света. Сви тестови спроведени су у развојном окружењу *PyCharm* помоћу модула *analysis* (4.3.3), док су слике коришћене за тестирање преузете из *Google Images* базе података [15].

Превасходно ћемо испитати рад приложених алгоритама на синтетичким сликама, и то оним које немају градијентно обојене површи, већ су уочљиви региони већином монотони. На слици 26 налази се приказ резултата рада Зановог алгоритма на дигитално конструисаној слици сове (а). Ова илустрација има за циљ да прикаже промену понашања алгоритма у зависности од растуће промене улазних параметара $c \in \{1, 2.6, 4.2, 5.8, 6.4, 8.0\}$ и $f \in \{2, 3.6, 5.2, 6.8, 7.4, 9.0\}$, док је $d = 3$ за све дате случајеве.



Слика 26. Пример рада Зановог алгоритма на синтетичкој слици димензија 417x626 за различите вредности параметара *c* и *f*

Параметри се мењају растуће, с лева на десно и одозго надоле. Уочљиво је да се са њиховим порастом заиста смањује број препознатих компонената у сегментацији, што је у

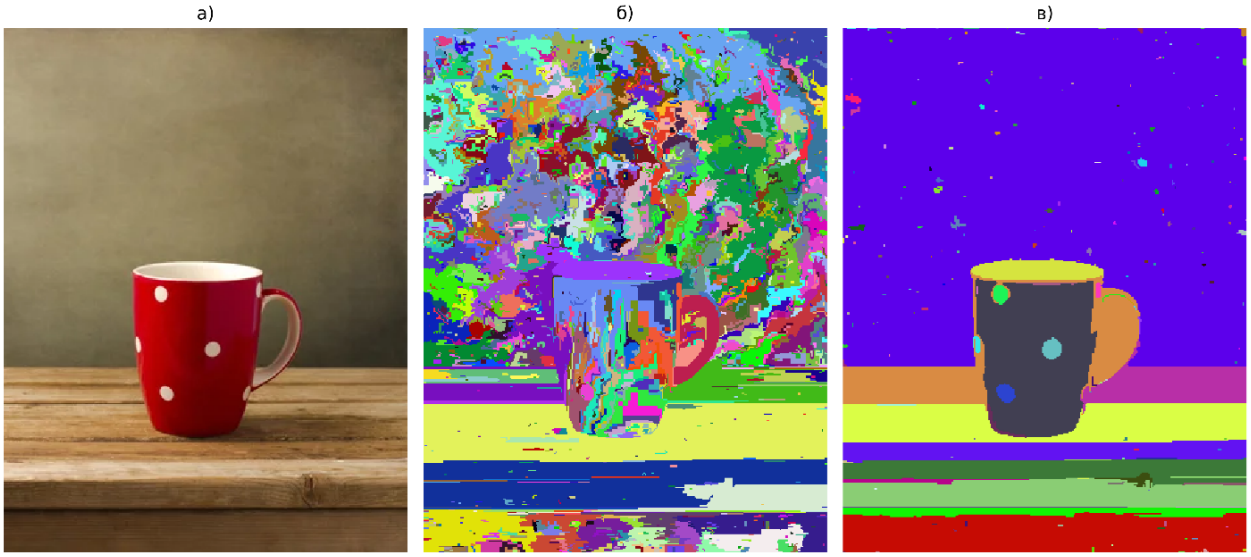
складу са теоријским разматрањем значења ових параметара. Са претходне слике могуће је закључити и да је одређену задовољавајућу сегментацију могуће постићи бирањем вредности које спадају у опсеге између вредности употребљене за случајеве г) и д). Ипак, заједничко за све случајеве јесте присуство великог броја малих региона, који на коначном приказу сегментације стварају ефекат шума. Ова појава је типична за Занов алгоритам у деловима слике у којима интензитети пиксела изразито варирају. Ако сада користећи исту улазну слику спроведемо Фелзеншвалбов алгоритам са растућим поретком вредности улазног параметра $k \in \{500, 900, 1300, 1700, 2100\}$, добијамо резултате на слици 27:

Слика 27. Пример рада Фелзеншвалбовог алгоритма на синтетичкој слици димензија 417x626 за различите вредности параметра *k*

Овде је такође приметно да се са повећањем улазног параметра смањује број региона, а самим тим и повећава њихова величина. Још једна сличност са претходним примером је постојање уских региона на границама између великих компонената, премда је та појава доста мање изражена у овом примеру. Ипак, из тог разлога је утисак да Фелзеншвалбов алгоритам даје квалитетнију сегментацију. Својеврстан проблем у раду ових алгоритама свакако је незаобилазан процес испробавања различитих вредности улазних параметара, не

би ли се добила задовољавајућа сегментација, што може бити изазовно. Избор ових параметара врши се на основу анализе саме улазне слике, као и на основу процене жељеног нивоа грануларности сегментације.

У наставку ћемо се позабавити радом разматраних алгоритама на стварним фотографијама. Најзначајније разлике између синтетичких и стварних фотографија огледају се у осветљености, обојености и количини детаља. Може се закључити да се стварне фотографије најчешће не састоје из монотоно обојених региона, већ да су уочени региони углавном градијентно обојени, уз присуство осенчености. Примећене особине стварних слика додатно усложњавају посао алгоритама сегментације. Следећи пример упоредно илуструје рад разматраних алгоритама на стварној слици, где је у питању јасно уочљива шоља на столу. Демонстрација је приказана на слици 28.

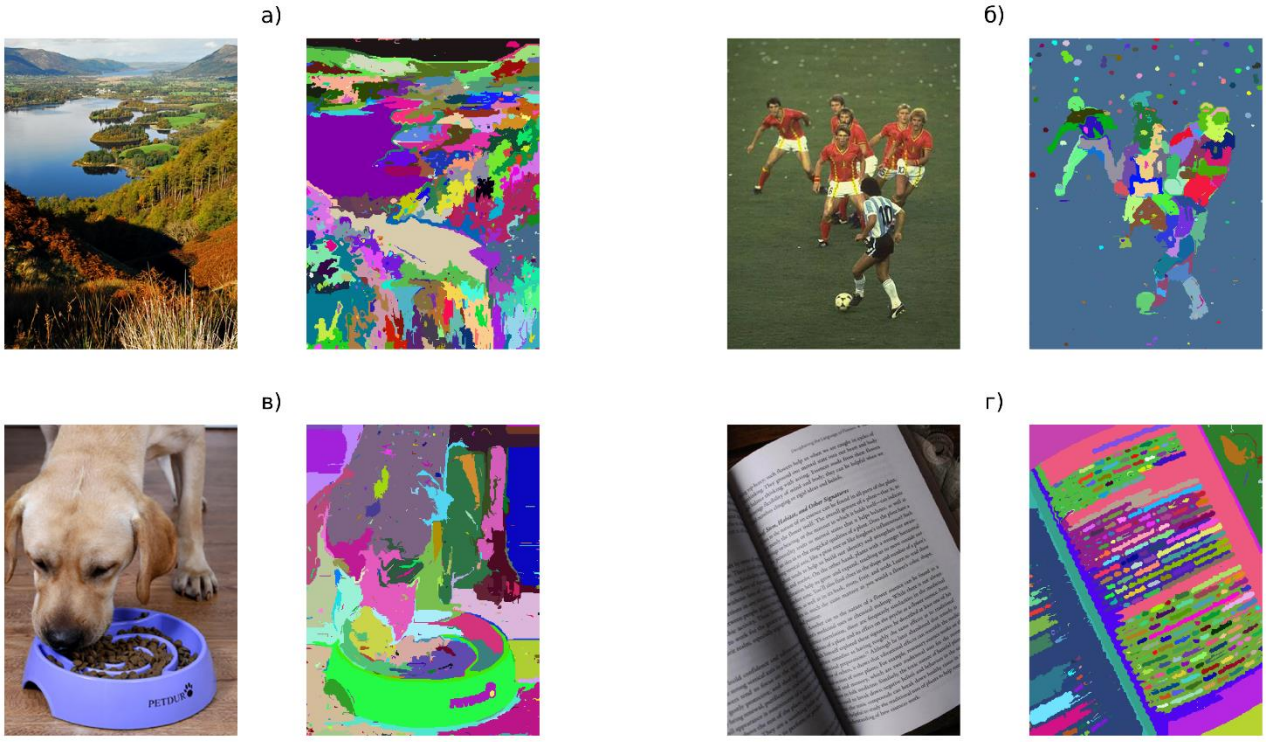


Слика 28. Пример сегментације стварне слике 308x400 користећи Занов алгоритам ($d = 3, c = 10, f = 10$) и Фелзеншвалбов алгоритам ($k = 4000$)

Овај пример веома је значајан, јер он истиче суштинску разлику између Зановог (б) и Фелзеншвалбовог (в) алгоритма. Могуће је уочити проблематику са радом Зановог алгоритма која се испољава у ситуацијама када се сегментира слика чија се текстура постепено мења. Критеријум овог алгоритма делује исувише локално, те, примера ради, зид који се налази иза шоље дели у велики број региона, зато што детектује разлике у интензитету обојености различитих делова зида. Са друге стране, Фелзеншвалбов алгоритам превазилази овај проблем и правилно уочава зид као јединствен регион (наравно, уз одређена

мања одступања) услед адаптивности свог критеријума, који правилно препознаје да се између два дела високе варијабилности не може једноставно повући граница.

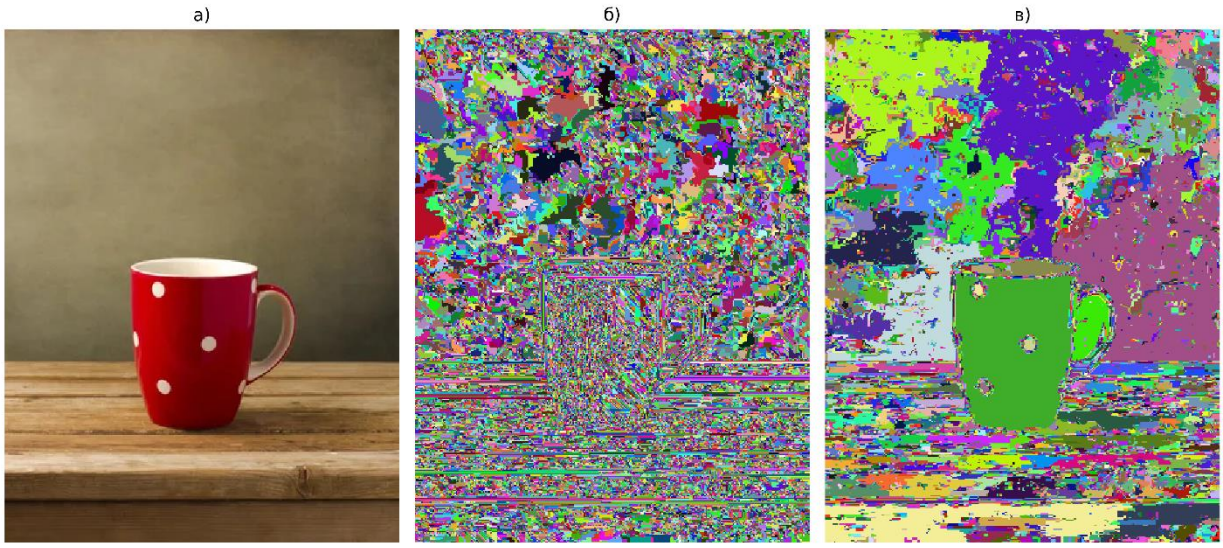
Наредни пример илуструје рад Фелзеншвалбовог алгоритма на сликама различитих интензитета боја, стила обојености и осветљења. Такође, слике се разликују и по детаљности, те је стога било важно адекватно одабрати параметар k ради квалитетне сегментације.



Слика 29. Сегментација стварних слика 300x400 користећи Фелзеншвалбов алгоритам, где су у питању фотографије: а) природе; б) фудбалске утакмице; в) пса који једе; г) отворене књиге

На слици 29 је приметно да фотографије природе и фудбалске утакмице обилују важним детаљима (шумовити предели, дресови фудбалера), те ту треба препознати потребу за прецизнијом сегментацијом и мањом вредности параметра $k = 500$. За сегментацију фотографије пса употребљена је нешто већа вредност $k = 1300$, док је фотографија отворене књиге релативно једноставна и не изискује превише фину грануларност сегментације, те је у том случају употребљена вредност $k = 4300$.

Сви примери који су до сада обрађивани користили су разматране алгоритме у монохроматском режиму. Последњи пример у овом одељку испитује дејство Фелзеншвалбовог алгоритма у режиму у боји. Фотографија која се користи као улаз алгоритма је приказ шоље на столу са слике 28. Резултати су приказани на слици 30.



Слика 30. Пример сегментације стварне слике 308x400 користећи Фелзеншвалбов алгоритам у боји

Случај б) представља резултате покретања овог алгоритма на начин описан у одељку 4.3.1, где се узима опрезнији приступ и грана увршатава у коначну сегментацију само у случају да се она нађе у сегментацијама које се односе на сваку од боја засебно. Очигледно је да овај приступ није дао задовољавајуће резултате, с обзиром на изразито велики број препознатих региона, чак и поред веома велике вредности улазног параметра $k = 10^7$. Из овог разлога се покушало са другачијим решењем у случају в), где је узет оптимистички приступ, и где се грана укључује у финалну сегментацију ако се она пронађе у било којој сегментацији бојâ. Одавде је за очекивати да ће се шансе за спајање региона у финалној сегментацији значајно повећати, те се алгоритам мора спроводити са изразито малим вредностима улазног параметра. Тако је у претходном примеру у случају в) искоришћена вредност $k = 1$ и добијена сегментација која је свакако квалитетнија од случаја б), пошто је на њој могуће јасно препознати контуре шоље. Ипак, очигледно је да чак и ова сегментација поседује превелики број препознатих региона. Општи закључак је да за предложени модел мрежастог графа посматрани алгоритми дају боље резултате у монохроматском режиму.

Временска сложеност алгоритама сегментације слика у овом поглављу није детаљно разматрана, али је експериментално утврђено да Фелзеншвалбов алгоритам даје нешто боље перформансе у односу на Занов алгоритам.

5. Закључак

Овај мастер рад написан је са циљем да читаоцу приближи неке од најзначајнијих примена минималних обухватних стабала у контексту апроксимативних решења различитих проблема. Разматрана структура података представља драгоцен начин одржавања релација међу подацима због своје оптимизационе природе и прилагодљивости другим сродним проблемима. Минимално обухватно стабло може послужити као полазна тачка и важан градивни елемент формирања нових алгоритама, док са друге стране и алгоритми за његову конструкцију могу представљати основ за осмишљавање нових приступа неком проблему.

На почетку рада дат је адекватан теоријски увод у графове и минимална обухватна стабла, где су разматрани многи важни појмови који су у даљем раду пронашли своју употребу. Наведени су и образложени неки познати алгоритми конструкције MST, међу којима је издвојен и демонстриран Крускалов алгоритам, а потом је кратко било речи и о применама MST, где су се као значајне споменуле области комбинаторне оптимизације и таксономије.

Проблем трговачког путника (3) је препознат као важан проблем из области комбинаторне оптимизације, и он је у овом раду детаљно обрађен са становишта примене MST. Осим што су предложене одређене хеуристике за ограничавање оптималног решења, наведена су и одређена апроксимативна решења овог проблема која у свом раду користе MST, од којих далеко најзначајније постигнуће свакако представља Кристофидисов алгоритам, који постиже резултат лошији од оптималног за највише 50%.

Проблем сегментације слика (4) послужио је као интересантан пример из области таксономије, где је примена минималних обухватних стабала на први поглед неинтуитивна, али након краће анализе проблема и правилног моделовања сасвим природна и прихватљива. Разматрана су следећа два алгоритма која у свом дејству користе MST: Занов алгоритам, који је први коришћен у области таксономије, и Фелзеншвалбов алгоритам, који је увео иновативне кораке детекције региона са циљем повећања адаптивности алгоритма.

Из самог наслова овог рада јасно је да су сви разматрани проблеми без апсолутно тачних и прихваћених решења, што доводи до закључка да при њиховом решавању увек има простора за додатна истраживања и будући напредак. У поглављу 2.3.1 наведен је проблем Штајнеровог стабла, као још један значајан проблем из области комбинаторне оптимизације. Наговештено је да постоје начини апроксимације овог проблема уз коришћење MST, па детаљнија изучавања у том смеру засигурно представљају природан корак у будућности. Код проблема трговачког путника може се покушати са оптимизацијом Кристофидисовог алгоритма, где је једна идеја убрзати алгоритам одређивања оптималног упаривања, а друга осмислити другачији начин постизања парних степена свих чворова у полазном MST. Разматрани алгоритми сегментације слика такође са собом носе одређене могућности побољшања. Један ток истраживања могао би бити усмерен ка паралелизацији ових алгоритама. С друге стране, могуће је покушати и са неким другим начином организације графа над којим се врши сегментација, те тада поново испитати дејство ових алгоритама. На пример, могуће је формирати граф где су чворови одређени векторима сачињеним од позиције и интензитета боја (енгл. *feature vectors*), а потом одређивати суседства на основу тих вектора. Тако би постојала могућност да се региони слике обојени истом бојом сврстају у исту компоненту, чак и ако они не представљају групације суседних пиксела.

6. Литература

[1] М. Томашевић, Алгоритми и структуре података, 2008, pp. 155-158, 172-179.

[2] J. Nešetřil, E. Milková и H. Nesetrilová, „Otakar Boruvka on minimum spanning tree problem; Translation of both the 1926 papers, comments, history,“ 2001.

[3] T. Cormen, C. Leiserson, R. Rivest и C. Stein, Introduction to algorithms, MIT, 2022.

[4] The University of Texas at Dallas, „Applications of minimum spanning trees“.

[5] F. K. Hwang, D. S. Richards и P. Winter, The Steiner Tree Problem, 1992.

[6] Wikipedia, „Travelling Salesman Problem,“ 2024. [На мрежи]. Available: https://en.wikipedia.org/wiki/Travelling_salesman_problem. [Последњи приступ фебруар 2024].

[7] E. Beresneva и S. Avdoshin, „The metric travelling salesman problem: pareto-optimal heuristic algorithms“.

[8] Reducible, „The Traveling Salesman Problem: When Good Enough Beats Perfect,“ 2022. [На мрежи]. Available: https://www.youtube.com/watch?v=GiDsjIBOVoA. [Последњи приступ фебруар 2024].

[9] MIT, „R9. Approximation Algorithms: Traveling Salesman Problem,“ 2015. [На мрежи]. Available: https://ocw.mit.edu/courses/6-046j-design-and-analysis-of-algorithms-spring-2015/. [Последњи приступ фебруар 2024].

[10] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, 1976.

[11] D. Johnson и L. McGeoch, The Traveling Salesman Problem: A Case Study in Local Optimization, 1995.

[12] G. Reinelt, „TSPLIB 95,“ 1991.

[13] V. K. G. K. Santle Camilus, „A Review on Graph Based Segmentation,“ MECS, 2012.

[14] Wikipedia, „Gestalt psychology,“ 2024. [На мрежи]. Available: https://en.wikipedia.org/wiki/Gestalt_psychology. [Последњи приступ фебруар 2024].

[15] Google, „Google Images,“ 2024. [На мрежи]. Available: https://images.google.com/. [Последњи приступ март 2024].

[16] D. P. H. Pedro F. Felzenszwalb, „Efficient Graph-Based Image Segmentation,“ MIT, 2004.

[17] C. T. Zahn, Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters, 1971.

[18] D. P. H. Pedro F. Felzenszwalb, „Image Segmentation Using Local Variation,“ Cornell University, 1998.

[19] М. Цветановић и З. Радивојевић, Како написати дипломски рад.

[20] Carnegie Mellon University, „One-Tree Relaxation,“ 1996.

[21] S. A. Nene, S. K. Nayar и H. Murase, „Columbia Object Image Library (COIL-100),“ 1996.