UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA

Stefana Janićijević

Metode promena formulacija i okolina za problem maksimalne klike grafa

DOKTORSKA DISERTACIJA

Novi Sad
2016

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Докторска дисертација
Аутор, **АУ**:		Стефана Јанићијевић
Ментор, **МН**:		Др Ненад Младеновић, научни саветник
Наслов рада, **НР**:		Методе промена формулација и околина за проблем максималне клике графа
Језик публикације, **ЈП**:		Српски
Језик извода, **ЈИ**:		Српски/Енглески
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2016
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Универзитет у Новом Саду, Факултет техничких наука
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/109/257/19/8/25/0
Научна област, **НО**:		Примењена математика
Научна дисциплина, **НД**:		Операциона истраживања
Предметна одредница/Кључне речи, **ПО**:		Максимална клика, Метода променљивих околина, Претрага кроз простор формулација, Модели, Глобална оптимизација, Велике базе података
УДК		
Чува се, **ЧУ**:		Библиотека Факултета техничких наука
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Докторска дисертација се бави темама решавања рачунарски тешких проблема комбинаторне оптимизације. Истакнут је проблем максималне клике као представник одређених структура у графовима. Проблем максималне клике и са њим повезани проблеми су формулисани као нелинеарне функције. Решавани су са циљем откривања нових метода које проналазе добре апроксимације решења за неко разумно време. Предложене су варијанте Методе променљивих околина на решавање максималне клике у графу. Повезани проблеми на графовима се могу применити на претрагу информација, распоређивање, процесирање сигнала, теорију класификације, теорију кодирања, итд. Сви алгоритми су имплементирани и успешно тестирани на бројним различитим примерима.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	Др Драган Урошевић, научни саветник
	Члан:	Др Татјана Давидовић, виши научни сарадник
	Члан:	Др Наташа Сладоје Матић, ванредни професор
	Члан:	Др Тибор Лукић, доцент
	Члан, ментор:	Др Ненад Младеновић, научни саветник

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		Monograph type
Type of record, **TR**:		Printed text
Contents code, **CC**:		PhD thesis
Author, **AU**:		Stefana Janićijević
Mentor, **MN**:		Nenad Mladenović, Ph.D., Research Professor
Title, **TI**:		Variable Formulation and Neighborhood Search Methods for the Maximum Clique Problem in Graph
Language of text, **LT**:		Serbian
Language of abstract, **LA**:		Serbian/English
Country of publication, **CP**:		Serbia
Locality of publication, **LP**:		Vojvodina
Publication year, **PY**:		2016
Publisher, **PB**:		Author's reprint
Publication place, **PP**:		University of Novi Sad, Faculty of Technical Sciences
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/109/257/19/8/25/0
Scientific field, **SF**:		Applied Mathematics
Scientific discipline, **SD**:		Operations Research
Subject/Key words, **S/KW**:		Variable Neighborhood Search, Formulation Space Search, Maximum Clique Global Optimization, Models, Big Data
UC		
Holding data, **HD**:		The Library of the Faculty of Technical Sciences, Novi Sad
Note, **N**:		
Abstract, **AB**:		This Ph.D. thesis addresses topics NP hard problem solving approaches in combinatorial optimization and according to that it is highlighted maximum clique problem as a representative of certain structures in graphs. Maximum clique problem and related problems with this have been formulated as non linear functions which have been solved to research for new methods and good solution approximations for some reasonable time. It has been proposed several different extensions of Variable Neighborhood Search method. Related problems on graphs could be applied on information retrieval, scheduling, signal processing, theory of classi_cation, theory of coding, etc. Algorithms are implemented and successfully tested on various different tasks.
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		
Defended Board, **DB**:	President:	Dragan Urošević, Ph.D., Research Professor
	Member:	Tatjana Davidović, Ph.D., Senior Research Fellow
	Member:	Nataša Sladoje Matić, Ph.D., Associate Professor
	Member:	Tibor Lukić, Ph.D., Assistant Professor
	Member, Mentor:	Nenad Mladenović, Ph.D., Research Professor

Menthor's sign

Zahvalnost

Ovom prilikom bih se zahvalila svima koji su mi tokom studija pomagali da učenje, istraživanje, programiranje, pisanje i pripremu doktorske disertacije dovedem do kraja, a prvenstveno da naučim, shvatim i zakoračim u veliki svet nauke i izazova u rešavanju teških problema današnjice. Pre svega zahvaljujem se mentoru, profesoru dr Nenadu Mladenoviću na prihvatanju mene kao studenta, na savetima, podučavanju, pomoći u vidu razumevanja optimizacije, putokazima i podršci, profesorima dr Tatjani Davidović i dr Draganu Uroševiću, na stručnoj pomoći, na konsultacijama, sugestijama i doprinosima, zatim profesorima dr Nataši Sladoje-Matić i dr Tiboru Lukiću koji su dali korisne sugestije i primedbe, zatim dr Milanu Dražiću na konsultacijama, doprinosima, idejama, profesorima dr Zoranu Markoviću i dr Zoranu Ognjanoviću, na ponudi za izvrstan posao istraživača u Matematičkom institutu SANU, bez koga ja ne bih došla do doktorata iz matematike. Zatim, moram se zahvaliti svojim prijateljima, istraživačima, zaljubljenicima u matematiku, programiranje, optimizaciju i metaheuristike, Radovanu Obradoviću i Kristini Veljković, bez čijeg prijateljstva, slušanja, razgovora o teoremama, strpljenja i nesebične stručne pomoći u programiranju ne bih mogla. Korisne savete u pogledu hipoteza, alata, programiranja i optimizacije algoritama dugujem naučnim istraživačima dr Peter Korošecu i dr Gregor Papi, čiji sam gost bila 10 meseci na institut Jožef Štefan u Ljubljani i uz koje sam dosta naučila i dosta savladala tehnike algoritama i programiranja. Pored toga, ideje o hipotezama i testiranjima takodje su potekle u našim zajedničkim razgovorima o ovoj tezi. Da nije bilo profesora dr Riste Skrekovskog, mog mentora na Fakultetu za Matematiku Univerziteta u Ljubljani ne bi bilo ni saradnje sa institutom Jožef Štefan, pa mu se ovom prilikom takodje zahvaljujem na tome kao i na lepom prijemu, poznanstvima, konsultacijama i seminarima na kojima sam učestvovala. Izuzetno se zahvaljujem fondaciji, odnosno administraciji i koordinatorima sa projekta Erasmus Mundus programa, zahvaljujući kome sam se i ja našla medju uspešnom i naprednom zajednicom mladih ljudi koji grade sopstvenu viziju boljeg života ispunjenog učenjem, razmenom iskustva, putovanjima i širenjem kulturoloških vidika. Osim toga, zahvaljujući predivnom iskustvu u Ljubljani, postala sam deo tima koji radi za dobrobit i poboljšanje uslova života svako u svojoj matičnoj zemlji i svako u svom profesionalnom okruženju. Pored njih, zahvaljujući profesoru Mladenoviću stupila sam u profesionalni kontakt sa poznatim naučnicima dr Jack Brimbergom sa Royal Military College u Kanadi i dr Sergejem Butenkom, Faculty Fellow in Industrial and Systems Engineering sa Univerziteta u Floridi, pa tako sa dr Brimbergom sam ostvarila rad na temu particije klike koji je predstavljen u tezi. Kada se osvrnem iza sebe i pogledam moje doktorske studije mogu da kažem da sam potpuno profesionalno bila ispunjena njima, upozala sam se sa istraživačima i naučnicima iz raznih oblasti, učestvovala sam na konferencijama i seminarima, učila sam iz zanimljive i prestižne naučne literature i bila upućena u najnovija dostignuća, pa sa tim u vezi, moram se zahvaliti profesorima i administrativnom osoblju Fakulteta tehničkih nauka, što sam postala deo njihovog tima i što su profesionalno i tačno vodili računa zajedno samnom o dostignućima, izazovima, rokovima i izveštajima koji su se ticali mojih doktorskih studija. Ne smem da preskočim niti profesore i osoblje Matematičkog instituta SANU gde sam napravila prve naučne korake i zavolela matematiku na jedan dublji i sigurniji način nakon diplomiranja na Matematičkom fakultetu u Beogradu. Hvala na otvorenosti i viziji o interdisciplinarnosti medju raznim oblastima u kojima sam se kretala u životu, a to su umetnost i nauka. S tim u vezi, zahvljajujem se i profesorima sa Fakulteta dramskih umetnosti koji su me svojim idejama usmeravali ka toj interdisciplinarnosti. Moram se zahvaliti i kolegama u kompaniji Telekom Srbija u kojoj radim i profesionalno se dokazujem u oblasti dejta majninga, da nije bilo izazova i realnih zadataka čije je rešenje dovelo do mnogih uspešnih kampanja danas, ne bih ni ja dobila viziju primene matematičke teorije na velike skupove podataka. Na kraju, najviše moram zahvaliti svojim roditeljima i bakama na tome što su uvek bili tu za mene, na tehničkoj i materijalnoj pomoći koju su mi pružali, i svom sinu Jovanu, koji mi je vraćao osmeh na lice u trenucima padova i podsećao da su ljubav i porodica osnova života.

Skraćenice korišćene u tezi

O	-	Bachmann-Landau or Asymptotic Notation
MC	-	Maximum clique
MVWC	-	Maximum vertex weighted clique
LP	-	Linear programming
NP	-	Nondeterministic polynomial time
P	-	Polynomial time
CPU	-	Central processing unit
SAS	-	Statistical analysis system
SGH	-	Sequential greedy heuristics
MLS	-	Multistart local search
GRASP	-	Greedy randomized adaptive search procedures
TS	-	Tabu search
GA	-	Genetic algorithm
DIMACS	-	Center for Discrete Mathematics and Theoretical Computer Science
BHOSLIB	-	Benchmarks with Hidden Optimum Solutions for Graph Problems
VNS	-	Variable neighborhood search
FI	-	First improvement
BI	-	Best improvement
VND	-	Variable neighborhood descent
VNDS	-	Variable neighborhood decomposition search
SVNS	-	Skewed variable neighborhood search
RVNS	-	Reduced variable neighborhood search
GVNS	-	General variable neighborhood search
CGVNS	-	Continuous general variable neighborhood search
FSS	-	Formulation space search
SGVNS	-	Skewed general variable neighborhood search
VFS	-	Variable formulation search
VOS	-	Variable objective search
VSS	-	Variable space search
FEHO	-	Fast Exact Higher Order
FHHO	-	Fast Heuristics Higher Order
FSSHO	-	Formulation Space Search Higher Order
DC	-	Difference convex
GOC	-	Global optimality conditions
GSA	-	Global search algorithm
ES	-	Evolution strategy
CCC	-	Cubic clustering criterion
PC	-	Partition coefficient
CE	-	Classification entropy
PI	-	Partition index
DB	-	Davies Bouldin index

Abstract

This manuscript presents Dissertation for PhD degree of the doctoral program Mathematics in Engineering at Faculty of the Technical Sciences, at Department of Applied Mathematics in Engineering. In this Dissertation, were developed various approaches for solving NP hard problems in combinatorial optimization and according to that it was highlighted clique partitioning problem and maximum clique problem as representatives of certain structures in graphs. Maximum clique problem and related problems were formulated as non linear functions and were solved in order to discover new methods to find good solution approximations within reasonable time. There were proposed several different extensions of Variable Neighborhood Search method such as SVNS and GVNS and some variants of Formulation Space Search method. Search through formulation space or FSS is using alternative formulations of problem to determine better one among the solutions that correspond to the same objective function values in original formulation. Beside maximum clique problem, this method is solving complementary problems such as independent set and vertex covering. This Dissertation describes optimization problems in detail, methods of optimization, graph concept theory, formulations of some graph structures and specific metaheuristics we deal with, algorithm implementation and testing with results. Last part of dissertation is refered to possible applications of the considered problem. After basic analysis and modeling, algorithms are directly applied to solution searching in special graph constructions according to databases which is relevant contribution to the proces of data mining techniques and to variable relation searching in big data structures.

Rezime

Rukopis predstavlja Disertaciju doktorskog programa Katedre za matematiku na Departmanu za opšte discipline u tehnici, Fakulteta tehničkih nauka u Novom Sadu. U ovoj disertaciji, razvijeni su pristupi u rešavanju računarski teških problema kombinatorne optimizacije i u vezi sa tim, istaknuti su problemi particije klike i maksimalne klike kao predstavnike odredjenih struktura u grafovima. Problem maksimalne klike i sa njim povezani problemi su formulisani kao nelinearne funkcije i rešavani su sa ciljem otkrivanja novih metoda koje pronalaze dobre aproksimacije rešenja za neko razumno vreme. Predložene su varijante Metode promenljivih okolina, SVNS i GVNS, kao i Metoda promenljivih formulacija. Pretraga kroz prostor formulacija ili FSS koristi više formulacija istog problema kako bi se odredilo bolje rešenje. Pored problema maksimalne klike, metoda rešava i komplementarne probleme maksimalnog nezavisnog skupa i čvornog pokrivača. Problemi na grafovima se mogu primeniti na pretragu informacija, rasporedjivanje, procesiranje signala, teoriju klasifikacije, teoriju kodiranja, itd. Svi algoritmi su implementirani i testirani na brojnim primerima iz različitih oblasti. Predloženi metod particije klike je dobar u odnosu na druge postupke koji su objavljeni u literaturi. Disertacija u velikoj meri opisuje probleme optimizacije, metode optimizacije, pojmove iz teorije grafova, formulacije odredjenih struktura u grafovima, konkretne metaheuristike, implementaciju algoritama, testiranje, dok se poslednji deo disertacije odnosi na moguće aplikacije odredjenog problema. Nakon osnovne analize i modela, algoritmi su primenjivi na pretrage rešenja na grafovima koji su dobijeni iz realnih baza podataka. Time se daje doprinos tehnikama otkrivanja podataka (engl. Data mining).

Sadržaj

1

Uvod

Poslednjih decenija optimizacija se razvija veoma brzo i vrlo dinamično s obzirom da ide u korak sa razvojem novih tehnologija. Algoritmi i tehnike programiranja utiču na otkrića u oblasti teorijskog računarstva kao i obratno, te se u optimizaciji koriste dostignuća različitih disciplina. Stalan rast uticaja raznih interdisciplinarnih oblasti koje utiču na nju i menjaju je je trend u optimizaciji. Optimizacija je osnov za sve oblasti istraživanja u inžinjerstvu, medicini i nauci. Donošenje odluka se bazira na optimizacionim procedurama koje se uspešno primenjuju u širokom opsegu praktičnih problema iz bilo koje sfere ljudske aktivnosti, uključujući biomedicinu, telekomunikacije, istraživanje svemira, infrastrukturu, finansije, itd [96].
Problemi koji se studiraju u ovoj disertaciji ilustruju interdisciplinarni razvoj optimizacije s obzirom da se primenjuju u raznim oblastima. Cilj istraživanja je primena Metode promenljivih formulacija (Metode pretrage prostora formulacija) i Metode promenljivih okolina na probleme kombinatorne optimizacije, definisane preko kontinualnih formulacija. Problem koji se najčešće koristi je problem rešavanja maksimalne klike. Pojmovi maksimalne klike i maksimuma klike su definisani u Poglavlju 3 u odeljku Maksimalna klika. U ovom radu se prvo prikazuje teorija optimizacije i matematičkog programiranja kroz istorijski period, a zatim se prezentuje formalan opis problema, istraživanja i rezultata koji su dobijeni. Problem pronalaženja najboljeg i najgoreg je oduvek bio od interesa da se reši. Problem trgovačkog putnika [63] je jedan od najpoznatijih problema optimizacije, a prvi problemi matematičkog modeliranja su zabeleženi čak u antičkoj Grčkoj i smatraju se najznačajnijim otkrićima tog vremena. Aleksandrijski matematičar Heron je rešavao problem nalaženja najkraćeg rastojanja izmedju dve tačke. Ovaj rezultat je poznat kao Heronova teorema svetlosnog zraka koja se kasnije utemeljila u teoriju geometrijske optike [47]. U sedamnaestom veku problem nalaženja ekstremne vrednosti je postigao specijalnu važnost kada je služio kao jedna od motivacija u pronalaženju diferencijalnog kalkulusa. Nakon razvoja varijacionog računa, i teorija stacionarnih tačaka leži u modernoj matematičkoj teoriji optimizacije. Tokom drugog svetskog rata algoritmi optimizacije su se koristili da rešavaju vojne, logističke i operativne probleme. Vojne aplikacije su motivisale razvoj linearnog programiranja (LP) tako što su bile prilagodjene optimizacionim problemima sa linearnom funkcijom cilja i linearnim ograničenjima. Dantzig je 1947. godine otkrio simpleks metodu za rešavanje linearnih programa koji su se koristili u U.S. vazdušnim operacijama. Linearno programiranje je postalo najpoznatija i dosta proučavana tema optimizacije uopšte [14].
Linearno programiranje je specijalan slučaj bitnog i širokog optimizacijskog opusa koji se naziva matematičko modelovanje, pri čemu se optimizuje funkcija cilja koja je linearna, dok su ograničenja tipa linearnih jednakosti (nejednakosti). LP predstavlja relaksaciju kombinatornog problema i koristi se u rešavanju za dobijanje dobrih donjih/gornjih granica i smanjivanje jaza izmedju njih. LP obuhvata i diskretne i kontinualne probleme [101]. Svi problemi koji se analiziraju u ovom radu su suštinski problemi globalne optimizacije. U nelinearnoj optimizaciji koriste se nelinearne funkcije na dopustivom domenu koji je opisan kao skup nelinearnih jednakosti(nejednakosti). Fascinantno je kako su se povezale tehnike kombinatorne optimizacije i nelinearne optimizacije i stvorile su nove, bolje optimizovane tehnike. Kombinovanje tehnika za rešavanje kombinatornih problema sa nelinearnim optimizacijskim pristupima je važno s obzirom da se tako omogućuje alternativna tačka gledišta koja vodi do novih karakterizacija analiziranih problema. Ove ideje daju svež pogled na kompleksne teme i često omogućavaju otkrivanje netrivijalnih veza izmedju

problema za koje se čini da imaju različitu prirodu. Celobrojno ograničenje (engl. Integer constraint), gde neke ili sve x_i imaju celobrojne vrednosti, je u formi $x \in \{0, 1\}$ jednako nekonveksnom kvadratnom ograničenju $x^2 - x = 0$. Ova jasna tvrdnja dokazuje da je prisustvo nekonveksnosti, a ne celobrojnosti, ono što čini optimizacioni problem teškim [128]. Izvanredna veza izmedju kombinatorne i nelinearne optimizacije se može sagledati kroz ovu disertaciju, tako što se izmedju ostalog definiše i dokazuje nekoliko neprekidnih formulacija problema maksimalne klike. To navodi na razvoj efikasnog algoritma koji traži maksimalnu kliku sa odredjenim osobinama (npr. najveća kardinalnost), a koja je bazirana upravo na neprekidnim formulacijama. Kao rezultat napretka optimizacijske metodologije i kao rezultat poboljšanja računarskih mogućnosti, skala rešivih problema iz oblasti optimizacije konstantno raste. Mnogi složeni problemi optimizacije ne mogu biti rešeni putem tradicionalnih optimizacionih tehnika. Heurističke metode traže i pronalaze što bolja približna rešenja [96]. Mnoge klasične heuristike su bazirane na procedurama lokalne pretrage, koje se iterativno pomeraju ka boljem rešenju (ako ono postoji) u okolini trenutnog rešenja. Procedure ovog tipa se obično prekidaju kada je prvi lokalni optimum dostignut. Randomizacija i restart se koriste kako bi se prevazišao siromašan kvalitet lokalnih rešenja, mada je to često nedovoljno efikasno. Generalnije strategije poznate kao metaheuristike obično kombinuju pristup heuristika u smeru rešenja boljeg kvaliteta nego što je to putem lokalne pretrage. Heuristike i metaheuristike igraju ključnu ulogu u velikim, teškim, primenjenim, optimizacionim problemima. Kroz ovaj rad se testiraju nove, efikasne metaheuristike za maksimalnu kliku [192].
Praktične primene za probleme na grafovima su vrlo široke i zadiru praktično u sve realne zahteve koji se javljaju u industriji i privredi. Osnovne oblasti primene su pretraga informacija, analiza prenosa signala, teorija klasifikacije, rasporedjivanje, procesiranje slike, data mining, itd [192]. Više o pojedinačnim primenama videti u Poglavlju 6.

1.1 Organizacija disertacije

Rad je podeljen u 7 poglavlja: Uvod, Problemi i metode optimizacije, Osnove teorije grafova i formulacije problema maksimalne klike, Metode promenljivih okolina i formulacija, Heuristike za problem klike u grafu, Primene i Zaključak. Na početku prvog poglavlja se prikazuju uvodni pojmovi koji se koriste u disertaciji. Nakon toga, se postavljaju fundamentalne činjenice imajući u vidu razmatrane probleme.
Drugo poglavlje se sastoji od definicije NP klase kompleksnosti, zatim se daje definicija problema optimizacije, pregled dosadašnjih metoda u rešavanju problema optimizacije, konkretno opis egzaktnih metoda u odnosu na heuristike i metaheuristike i na kraju poglavlja se predstavljaju ukratko opisane odredjene uspešne metode koje se najčešće koriste u časopisima.
Treće poglavlje prikazuje osnovne pojmove teorije grafova, kao i definicije i pojmove konkretnog grafovskog problema koji se razmatra a to je maksimalna klika. Zatim je opisan problem maksimalne klike i formulacije za taj problem: osnovna kombinatorna, zatim celobrojna, mešovita, kvadratna, pa na kraju kontinualna i nekonveksna. Problem maksimalne klike je poznat kao problem sa preko deset aktivnih zvaničnih formulacija, ali kako istraživanja u numeričkoj analizi napreduju, vrlo je moguće povećati taj broj, posebno što se stalno izvode testiranja novih formulacija.
Četvrto poglavlje se sastoji od opisa i pregleda Metoda promenljivih okolina i formulacija, s obzirom da je istraživanje nastalo na osnovu razvoja pomenutih metaheuristika. Osnovna ideja Metode promenljivih okolina je jednostavna i sastoji se od sistematske promene okolina unutar lokalnog pretraživanja. Metoda je do sada uspešno primenjena na veliki broj problema kombinatorne optimizacije i kontinualne optimizacije. Iz godine u godinu raste broj modifikacija i poboljšanja, uglavnom namenjenih uspešnom rešavanju problema iz oblasti globalne optimizacije. U okviru ovog poglavlja predstavljaju se dva rada koja se bave problemom rasporedjivanja i na njima se radilo na početku istraživanja.
Zatim se definiše Metoda promenljivih formulacija generalno. To je još uvek relativno mlada metaheuristika koja je proizašla iz Metode promenljivih okolina, a kompleksnija je u odnosu na druge s obzirom da je neophodno poznavati ponašanja funkcija kojima se predstavljaju formulacije. S tim u vezi, neophodno je poznavati prostor rešenja svake funkcije, transformaciju iz jednog prostora rešenja u drugi, odnosno preslikavanje rešenja iz jedne formulacije u drugu i sve

to kombinovati u osnovni algoritam, pošto se pretpostavlja da se kombinovanjem raznih funkcija cilja istog problema dolazi do boljih heurističkih rešenja nego korišćenjem samo pojedinačnih funkcija.
Peto poglavlje predstavlja neka od istraživanja koja su u toku doktorskih studija obavljena a tiču se problema maksimalne klike. S tim u vezi, prvo se prikazuje algoritam Metode promenljivih okolina koji je primenjen na maksimalnu težinsku kliku. Implementacija algoritma je rezultirala rešenjima koja su solidna i koja su čak u odredjenim instancama postigla bolje rezultate nego što su dali ostali algoritmi, što se prikazuje na kraju tog odeljka. Sledeći odeljak u okviru petog poglavlja prikazuje problem particije klike, odnosno rešava se problem maksimalnog različitog grupisanja. Problem se rešava Adaptivnom metodom promenljivih okolina koja se pokazala najefikasnije u ovom slučaju. Rezultati koji su postignuti su "state-of-the-art" na problemima velikih dimenzija. Problem particije klike se direktno koristi u metodi klasterizacije, što je takodje predmet ovog istraživanja u primeni klike. Poslednji odeljak u ovom poglavlju se koncentriše na promene formulacija u okviru problema maksimalne klike. Na početku su razvijana četiri algoritma: prvi je na osnovu Evolucione strategije, drugi je na osnovu GLOB paketa, treći je na osnovu Generalne metode promenljivih okolina, a četvrti je pretraživanje formulacija neprekidnog tipa koje su date kao skup formulacija u L_p normi. Rezultati su dati u prilogu na kraju odeljka. Rezultati su dobri za male primere s obzirom da je formulacija funkcije jako teška da se reši na velikim primerima grafova.
Na kraju poglavlja, u poslednjem odeljku, kao posledica prethodnih testiranja, prikazane su tri uspešne metode koje su razvijene pri rešavanju problema maksimalne klike. Prva metoda predstavlja enumerativni algoritam u kom se pretraga veličine maksimuma klike vrši na osnovu analize čvorova koji imaju maksimalni stepen. To je Brzi egzaktni algoritam višeg reda (engl. Fast Exact Higher Order, FEHO). Druga metoda predstavlja heuristiku takodje razvijenu na osnovu maksimalnog stepena čvorova i naziva se Brzi heuristički algoritam višeg reda (engl. Fast Heuristics Higher Order, FHHO). Treća metoda predstavlja primenu Metode promenljivih formulacija i naziva se Metoda promenljivih formulacija višeg reda (engl. Formulation Space Search Higher Order, FSSHO).
Poslednje poglavlje pre Zaključka prikazuje Primene. Tu su predstavljeni opisi problema koji se oslanjaju na maksimalnu kliku i nad kojima je moguće primeniti algoritam. Na kraju poglavlja data je primena rešavanja klike Metodom promenljivih okolina na problem klasterovanja baze podataka. U istraživanju je uporedjena Metoda promenljivih okolina sa SAS/STAT procedurom za klasterovanje i takodje su dati rezultati. U okviru pojedinih odeljaka predstavljeni su izazovi klasifikacije podataka u velikim bazama. Predstavljene su tehnike optimizacije koje postoje u okviru programa koji se zvanično koriste. Time se čitalac upoznaje sa idejom takozvanih tržišnih grafova, odnosno sa bazama koje se koriste u privredi. Graf i baza podataka se povezuju na taj način što se intervalne i ostale numeričke varijable predstavljaju čvorovima, a ivice se konstruišu na osnovu informacije o povezanosti izmedju čvorova.
Poslednje poglavlje je Zaključak.

1.2 Doprinosi disertacije

Tokom doktorskih studija i istraživanja problema optimizacije, istakli su se problemi na grafovima tj. preciznije klike, kao bliski autoru. Klike su interesantne zbog svoje strukture, naizgled jednostavne definicije iza koje se kriju mnoga dostignuća, ali i zbog primene u životu, u tekućim problemima, u društvu, u socijalnim mrežama (opis problema i istraživanja su opisani u kasnijim poglavljima).
Temu su profilisale metode promene formulacija i promene okolina nad klikama, s obzirom da je algoritam Metoda promenljivih formulacija relativno novi i još uvek neistražen dovoljno. U okviru istraživanja nastali su sledeći radovi, dok su se u kasnijim poglavljima detaljno objasnila i opisala istraživanja na njima:

1. S. Janićijević, D. Urošević, N. Mladenović, Comparison of SAS/STAT procedure and Variable Neighbourhood Search based clustering applied on Telecom Serbia data, Symopis Conference, 2014. [134]

2. S. Janićijević, D. Urošević, N. Mladenović, Comparison of SAS/STAT procedure and Vari-

able Neighbourhood Search based clustering applied on Telecom Serbia data, ITIS Conference, 2014. [135]

3. S. Janićijević, Z. Lužanin, S. Pereverzyev, I. Stojkovska, A. Tepavčević, Credit score card for corporate clients based on industries, Working Group ESGI99, Novi Sad, 2014. [133]

4. J. Brimberg, S. Janićijević, N. Mladenović, D. Urošević, Solving the Clique Partitioning Problem as a Maximally Diverse Grouping Problem, Optimization Letters, 1-13, 2015. [32]

5. S. Janićijević, N. Mladenović, R. Obradović, D. Urošević, VNS for Maximum vertex weighted Clique Problem Approximation, Symopis Conference, 2015. [136]

6. S. Janićijević, N. Mladenović, R. Obradović, Fast Exact Higher Order Algorithm for Maximum Clique Problem, trenutno na recenziji [137]

7. S. Janićijević, N. Mladenović, R. Obradović, Fast Heuristics Higher Order Algorithm for Maximum Clique Problem, trenutno na recenziji [138]

8. S. Janićijević, N. Mladenović, R. Obradović, Formulation Space Search Algorithm Higher Order for Maximum Clique Problem, trenutno na recenziji [139]

Rezultati u Disertaciji su predstavljeni u dva dela. Prvi deo rezultata predstavlja grupu od tri nezavisna algoritma, a drugi deo rezultata predstavlja jedan rad koji pokazuje primenu klike na rešavanje klasterizacije velike baze podataka.
Prvi deo rezultata predstavlja tri nezavisna algoritma:

1. Rešavanje maksimuma težinske klike (VNS for Maximum Vertex Weighted Clique Problem Approximation)

2. Rešavanje particije klike (Solving the Clique Partitioning Problem as a Maximally Diverse Grouping Problem)

3. Rešavanje maksimuma klike (Higher Order Vertex Degree and FSS for Maximum Clique Problem Approximation)

Drugi deo rezultata predstavlja jedan algoritam:

1. Rešavanje klastera u velikoj bazi podataka (Comparison of SAS/STAT procedure and Variable Neighbourhood Search based clustering applied on Telecom Serbia data)

U prvom delu rezultata razvijeni su algoritmi za aproksimaciju maksimuma (težinske) klike zasnovani na VNS metaheuristici. Predstavljeni su rezultati za svaki od pomenutih problema.

1. Problem maksimuma težinske klike: razvijen je algoritam zasnovan na podalgoritmima u kojima su implementirane lokalne pretrage po fazama. Izbegnuta je stagnacija rešenja tako što je izvršena njihova perturbacija na sistematski način VNS metodom.

2. Problem particije klike: razvijen je algoritam koji je kombinacija Adaptivnog VNS metoda i Generalnog VNS metoda, dakle Adaptivno-Generalni metod VNS koji bolje pretražuje prostor rešenja, odnosno koristan je kako bi se izbegla stagnacija u lokalnim optimumima.

3. Problem maksimuma klike: razvijena je nova klasa algoritama koja koristi kontinualne formulacije problema. U postupak rešavanja kontinualne formulacije uvedeni su metodi koji se uobičajeno koriste pri rešavanju diskretnih formulacija (stepeni čvorova, stepeni čvorova višeg reda, penali, lokalna pretraga, itd.). Dodatna diversifikacija rešenja je ostvarena sistematskom promenom formulacija problema. Dobijeni su rezultati koji su uporedivi sa najboljim rezultatima objavljenim u literaturi. Pre implementacije FSS algoritma, implementirana su četiri testna algoritma: GVNS, GLOB VNS, Evolutivna strategija, Promenljive formulacije. Ovi algoritmi su testirani na malim grafovima i služe kao pionirske metode spremne za dalje razvijanje nad velikim grafovima. Brzo se izvršavaju, jednostavni su za implementaciju i za testirane primere su dali dobre rezultate.

U drugom delu rezultata razvijen je algoritam primenjiv na velike baze podataka.

1. Problem klasterizacije baze podataka: razvijena je metoda VNS koja meri funkciju dobitka/gubitka tačaka po klasterima, a takođe razvijena je k-mean procedura u SAS programskom jeziku pa su obe metode upoređivane. VNS algoritam se pokazao efikasnijim i to je izmereno parametrima: koeficijentom determinacije, kubičnim kriterijumom klastera i vizuelizacijom klastera. Rad je prvo prikazan na Symopis, a zatim na ITIS konferenciji gde je dodatak bio uvođenje evaluacije klastera, pa je s tim u vezi programiran i meren Davies–Bouldin indeks koji meri koliko je klasterovanje dobro izvršeno. Indeks je takođe prikazao VNS kao stimulativniju metodu.

2

Problemi i metode optimizacije

2.1 Uvod

Problem optimizacije se definiše na sledeći način [63]:
Dati su skup S i funkcija $f : S \to \mathbb{R}$. Potrebno je naći minimum funkcije f na skupu S, odnosno rešiti optimizacioni problem

$$\min_{x \in S} f(x) \tag{2.1}$$

skup S je **dopustivi skup**, funkcija f se naziva **funkcija cilja** a x je **dopustivo rešenje** problema 2.1. Ukoliko je skup S diskretan tada je to **kombinatorna** (diskretna) optimizacija, ukoliko je S kontinualan tada je to problem **kontinualne** (globalne, neprekidne) optimizacije. Potrebno je pronaći sva dopustiva rešenja x^* takva da je

$$f(x^*) = \min_{x \in S} f(x)$$

Takva rešenja se nazivaju **optimalna rešenja**.
Problem maksimizacije se svodi na problem minimizacije zato što važi

$$\max_{x \in S} f(x) = \min_{x \in S}(-f(x))$$

Problem se formuliše kao zadatak **matematičkog programiranja**, pa se onda rešava raznim algoritmima. Optimizacija se sastoji od maksimizacije ili minimizacije realne funkcije na osnovu sistematičnog izbora ulaznih vrednosti iz dopustivog skupa i izračunavanja vrednosti funkcije [96]. U slučaju **celobrojnog** programiranja S je podskup skupa $\mathbb{Z}^n$ gde je $\mathbb{Z}^n$ skup n-dimenzionih vektora sa celobrojnim koordinatama. Problem **linearnog** celobrojnog programiranja:

$$\min_{x \in S} c^T x, S = \{x \in \mathbb{Z}^n | Ax \leq b\} \tag{2.2}$$

u kome je A celobrojna matrica tipa $m \times n$, a c i b celobrojni vektori odgovarajućih dimenzija. Ukoliko se u 2.2 relaksira uslov celobrojnosti dobija se problem linearnog programiranja:

$$\min_{x \in X} c^T x, X = \{x \in \mathbb{R}^n | Ax \leq b\} \tag{2.3}$$

Rešavanje problema celobrojnog linearnog programiranja se zasniva na uzastopnim relaksiranostima uslova celobrojnosti.
Specijalan slučaj problema 2.2 jeste problem $0-1$ programiranja, tj:

$$\min_{x \in S} c^T x, S = \{x \in \mathbb{B}^n | Ax \leq b\} \tag{2.4}$$

gde je $B = \{0, 1\}$

Problem **kvadratnog** programiranja se svodi na:

$$\min \frac{1}{2} x^T Q x + c^T x \quad pod\ ogranicenjem\ \ (p.o.) \quad Ax \leq b \tag{2.5}$$

gde je problem kvadratnog programiranja dat sa n promenljivih i m ograničenja tako da je c realni, n-dimenzioni vektor, Q realna, simetrična matrica dimenzija $n \times n$, A realna matrica dimenzija $m \times n$, b realan vektor dimenzije m, a x nepoznati n-dimenzioni vektor. Notacija $Ax \leq b$ pokazuje linearno ograničenje, dok se kvadratno ograničenje dobija uvodjenjem kvadratne funkcije na tom mestu.
Najčešće, globalna optimizacija je veoma teška zbog postojanja mnogo lokalnih minimuma i taj broj teži da raste eksponencijalno sa dimenzijom problema. Moguće je dizajnirati metodu koja omogućava da se u ϵ okolini pronadje globalni minimum. Mnogo je metoda za globalnu optimizaciju predloženo, neki od njih su deterministički a neki nedeterministički [128]. Osnovne metode u okviru determinističke strategije jesu unutrašnja i spoljašnja aproksimacija (engl. Inner and Outer approximation). Skup na kome se funkcija optimizuje se aproksimira nekim poliedrom kod obe metode.

2.2 NP problemi

U teoriji kompleksnosti postoji mnogo klasa koje se mogu definisati resursima koje algoritam koristi kako bi rešio odredjeni problem. Resursi mogu biti vremenski i memorijski. Neke od poznatijih klasa problema odlučivanja su: P, NP, BPP, ZPP, EXPTIME, NEXPTIME, PSPACE, NPSPACE, APX, PTAS, itd.
NP klasa problema spada u računarski rešive zadatke, poput P klase problema. Pored računarski rešivih zadataka, postoje i računarski nerešivi zadaci koji se u bilo kom poznatom računarskom okruženju ne mogu rešiti. Algoritamski rešivi zadaci su oni za koje postoji algoritam primenjiv na svaki individualni zadatak tog zadatka. Za razliku od računarski nerešivih zadataka klasa računarski rešivih zadataka se razlikuje po složenosti, odnosno kompleksnosti. To znači da je potrebno izmeriti resurse procesorskog vremena potrebnog za nalaženje rešenja i količinu memorijskog prostora koji je iskorišćen u rešavanju. U odnosu na potrebno vreme, algoritamski rešivi zadaci se dele na potklase. Neke od najpoznatijih potklasa su oni zadaci koji se mogu rešiti u polinomijalnom vremenu (P klasa) i oni zadaci koji se mogu rešiti u eksponencijalnom vremenu (NP klasa).
NP klasa je skup problema odlučivanja koji su rešivi u polinomijalnom vremenu na nedeterminističkoj Tjuringovoj mašini, ili, to je skup problema čija rešenja mogu da se verifikuju na determinističkoj Tjuringovoj mašini u polinomijalnom vremenu. Problem jeste NP ako se za njega može slučajno izabrati potencijalno rešenje i ako se determinističkim polinomskim algoritmom može proveriti da li je to pravo rešenje. NP znači Nondeterministically Polynomial, što znači da se problemi rešavaju pomoću nedeterminističkih polinomskih algoritama. Glavna razlika klasa P i NP je u slučajnom izboru potencijalnog rešenja. Algoritmi za probleme iz klase NP dobijaju svojstvo nedeterminisanosti. Pitanje koje je otvoreno i teško da se reši jeste da li je klasa P prava potklasa klase NP, kao što se intuitivno očekuje. Neki od osnovnih problema iz klase NP su Problem trgovačkog putnika, Problem izomorfizma grafova, Problem zadovoljivosti (engl. Boolean satisfiability problem, SAT) problem, Problem testiranja složenosti. Svaki zadatak koji se može rešiti na računaru odredjen je spiskom parametara i opisom svojstava koje mora da zadovoljava rešenje zadatka. Kada parametri dobiju neke konkretne vrednosti dobija se individualni zadatak. Potrebno je da algoritam bude efektivan. Jedan od velikih izazova u okviru istraživanja u optimizaciji jeste pronalaženje algoritama za probleme u NP klasi koji rade u polinomijalnom vremenu. Pri merenju efektivnosti algoritama koristi se simbol O (veliko O). Vremenska složenost nekog algoritma predstavlja maksimalno vreme izvršavanja algoritma, npr. zadatak dimenzije n predstavlja količinu elementarnih operacija koje algoritam izvršava, dok elementarna operacija zahteva fiksiranu količinu vremena. Za najbolju ocenu vremenske složenosti odredjuje se funkcija $g(n)$ sa najsporijim rastom uz povećanje vrednosti broja n. Vremenska složenost algoritama pretrage vrednosti u neuredjenom nizu sa n elemenata iznosi $O(n)$, a u uredjenom nizu iznosi $O(log(n))$, dok kod algoritama sortiranja iznosi $O(nlog(n))$. Za neke probleme vreme potrebno za izvršavanje algoritma raste eksponencijalno sa dimenzijom problema, dok za neke to vreme raste sporije. Problemi sa eksponencijalnom vremenskom složenosti se nazivaju teško rešivim problemima, dok su problemi sa polinomskom vremenskom složenosti pogodni za rešavanje na računarima.
Problem za koji se rešenje dobija u obliku DA-NE jeste problem odlučivosti. Ako je rešenje

nekog problema optimizacije $\min f(x) = MIN$ tada se on formuliše kao problem odlučivosti: *Za svako x, $f(x) \geq MIN$, postoji x_1 tako da važi $f(x_1) = MIN$.* Problem odlučivosti pripada klasi P ako se rešava determinističkim algoritmom sa polinomskim vremenom složenosti. Deterministički algoritam se predstavlja klasičnom Tjuringovom mašinom ili Postovom mašinom ili pak Markovljevim algoritmom. Deterministički algoritmi se zaustavljaju posle konačnog broja koraka za svaku kombinaciju ulaznih parametara. Problemi kombinatorne optimizacije se rešavaju pretragom, tako što se ispituje za svako potencijalno rešenje da li predstavlja i stvarno rešenje. Ispitivanje jednog potencijalnog rešenja se obavlja putem polinomskog algoritma. Za problem trgovačkog putnika rešenje se može dobiti tako što se odrede svi putevi koji počinju u gradu g_1, za svaki se odredi dužina i izabere se najkraći. Ako je broj gradova n, onda je broj puteva $(n-1)!$, što znači da bi se pretraga dugo izvršavala. Problemi koji se rešavaju kompletnom pretragom zahtevaju ispitivanje mnogo mogućnosti. Tako se dolazi do pomenute klase problema NP gde se vreme složenosti može izraziti polinomskom funkcijom, ali su algoritmi znatno složeniji.
Problem B je svodljiv na problem C akko postoji algoritam A kojim se svaki individualni problem IB problema B svodi u polinomijalnom vremenu na individualni problem IC problema C. Postoji još definicija svodljivosti u dosadašnjoj literaturi [216]. Klasifikacija problema se dobija svodjenjem problema.

Lema 2.2.1 *Ako problem B pripada klasi P i problem A može da se svede na B, onda i problem A pripada klasi P. Slično, ako problem A pripada klasi NP i problem A može da se svede na problem B onda i problem B pripada klasi NP.*

Time se u klasi NP izdvaja jezgro najtežih problema koji se nazivaju NP-kompletni problemi. Problem A je NP-kompletan ako pripada klasi NP i svaki drugi problem B iz NP je polinomski svodljiv na problem A.
Lema o svodljivosti govori da NP-kompletni zadaci predstavljaju najteže zadatke u klasi NP. Ako za bar jedan NP-kompletan problem postoji deterministički algoritam sa polinomskom složenošću, tada bi to važilo i za sve zadatke iz klase NP. Ako je bar jedan NP-kompletan problem teško rešiv onda to važi za sve NP-kompletne probleme. Pitanje je da li postoji bar jedan problem koji je NP-kompletan. Jedan od prvih problema za koji je dokazano da je NP-kompletan je problem iz matematičke logike, problem zadovoljivosti:
Dat je logički izraz $F(x_1, x_2, \ldots, x_n)$ u konjuktivnoj normalnoj formi. Da li postoji n-torka $(u_1, u_2, \ldots, u_n)$ tako da je $u_i \in \{\top, \bot\}$ i da je $F(u_1, u_2, \ldots, u_n) = \top$. Ovaj problem se rešava potpunom pretragom. Promenljivima $x_1, x_2, \ldots, x_n$ se dodeljuju vrednosti iz skupa $\{\top, \bot\}$ i formira se ukupno 2^n različitih n-torki. Za svaku od n-torki proverava se da li je vrednost izraza F jednaka $\top$, rešava se problem zadovoljivosti i tako on postaje teško rešiv problem.
Kuk je formirao teoriju NP-kompletnosti [60] 1971. godine i pokazao je da je zadatak zadovoljivosti NP-kompletan. Nakon toga za veliki broj problema je pokazano da predstavljaju NP-kompletne probleme.
APX ili Aproksimativna klasa (engl. Approximable class) i ona predstavlja skup NP optimizacionih problema koji se rešavaju pomoću aproksimativnih polinomijalnih algoritama čiji je aproksimativan faktor ograničen konstantom (engl. Constant-factor approximation algorithm). PTAS ili Polinomijalna aproksimativna shema (engl. Polynomial time approximation scheme) je klasa koja se sastoji od problema koji se mogu aproksimirati bilo kojim konstantnim faktorom osim jedinice za vreme koje polinomijalno izračuna taj konstantan faktor. PTAS je podskup klase APX.
Problem maksimalne klike se koristi u računskoj teoriji kompleksnosti a potiče iz teorije grafova i spada u Karpove NP kompletne probleme [91]. Karpov je u radu o svodjivosti medju kombinatornim problemima, 1972. godine dokazao polinomijalno-vremensku svodljivost iz problema SAT na 21 grafovski problem medju kojima je bio i problem klike, a pokazao je i da su svi ti problemi NP kompletni. Nadalje, problem maksimalne klike jeste jedan od prvih problema za koji je pokazano da je NP-kompletan [82], što znači da za egzaktne algoritme važi da računaju rešenje u vremenu koje eksponencijalno raste sa brojem čvorova u grafu [91]. Maksimum stabilnog skupa i minimum čvornog pokrivača su takodje NP kompletni.
Crescenzi, Fiorini i Silvestri [62] su dokazali da su svi problemi u $MAXNP$ jako reducibilni na problem maksimum klike. Kao posledica, ako je maksimum klike aproksimativan sa konstantnim

faktorom, onda su svi problemi u $MAXNP$ aproksimativni sa bilo kojim malim faktorom. Ovo dokazuje da problem maksimuma klike ne sadrži aproksimativan algoritam polinomijalne vremenske složenosti. Jači dokaz ove činjenice je dat nezavisno u okviru iste godine, kada su Feige i drugi [82] dokazali da ako postoji algoritam polinomijalne vremenske složenosti koji aproksimira problem maksimuma klike sa faktorom od $2^{\log^{1-\epsilon} n}$ onda bilo koji NP problem može biti rešen u "kvazi polinomijalnom" vremenu (npr. $2^{\log^{\mathcal{O}(1)} n}$).
Pokazano je da ni jedan polinomijalno vremenski algoritam ne može aproksimirati veličinu maksimum klike sa faktorom od n^{ϵ}, ukoliko nije $P = NP$, koristeći [82]. Radovi Feiga i drugih, Arore i drugih [13, 82] stimulisali su mnogo istraživanja na temu odnosa aproksimacije algoritma ali su dobijeni slabi rezultati. Najbolji aproksimativni algoritam polinomijalno vremenske složenosti za problem maksimuma klike su razvili Boppana i Halldorsson [30] i dostigli su aproksimativni odnos od $n^{1-o(1)}$. Hastad [120] je pokazao da je ovo najbolje što se može dostići. Iako ovi rezultati kompleksnosti karakterišu slučajeve najgorih instanci, oni indukuju da je problem maksimuma klike veoma težak da se reši. Ukoliko se posmatraju specijalne strukture grafova, tada se u mnogim slučajevima problem maksimum klike/stabilnog skupa može rešiti u polinomijalnom vremenu. Balas i drugi [16] su predstavili nekoliko klasa grafova i pokazali da se problem maksimuma težinske klike može rešiti u polinomijalnom vremenu. Balas i Yu [15] diskutuju klase grafova koji imaju polinomijalno mnogo maksimalnih klika. Na ovim grafovima, problem maksimuma težinske klike se može rešiti u polinomijalnom vremenu.

2.3 Podela metoda optimizacije

Metode optimizacije se dele na egzaktne, (meta)heurističke i simulacione [128]. Prva grupa je najpouzdanija ali je njihova primena na probleme većih dimenzija skoro nemoguća. Treća nije pouzdana, ali broj zadataka, koji pomuću te grupe može biti rešen jeste najveći. Drugu grupu čine algoritmi koji se baziraju na efikasnim procedurama i daju uglavnom približna rešenja. Heuristike predstavljaju konačan skup koraka kojima se dobijaju rešenja problema kombinatorne optimizacije (bez garancije njihove optimalnosti) za relativno kratko vreme. Osnovna prednost heurističkih metoda je njihova brzina, što omogućava dobijanje zadovoljavajućih rešenja za probleme velikih dimenzija kakvi se najčešće javljaju u realnim primenama. Kako su se heuristike pokazale kao praktično jedini način rešavanja optimizacionih zadataka, istraživači u poslednje vreme sve više pažnje posvećuju njihovom razvoju i usavršavanju. To je dovelo i do razvoja generalnih heuristika ili tzv. metaheuristika [192]. Klasične heuristike su, bile namenjene rešavanju nekih konkretnih, pojedinačnih problema i koristile su poznate osobine datog problema pri njegovom rešavanju. Metaheuristike, se sastoje od uopštenih skupova pravila koja se mogu primeniti za rešavanje raznovrsnih problema optimizacije. Metaheuristički pristupi u rešavanju optimizacionih problema zasnovani su na opštim algoritmima optimizacije koji podrazumevaju primenu iterativnih postupaka u cilju popravljanja nekog postojećeg rešenja. Standardne heuristike se razvijaju za jedan konkretan problem. Metaheuristika se sastoji od pravila i načela koja mogu biti primenjena na veliki broj različitih problema. Primenom opštih metodologija na konkretan problem dobija se heuristički princip rešavanja problema [96].
(Meta)heuristike se dele na sedam tipova:

1. Konstruktivne heuristike
2. Heuristike iterativnog poboljšanja
3. Heuristike matematičkog programiranja
4. Dekompozicione heuristike
5. Podela dopustivog skupa
6. Restrikcija dopustivog skupa
7. Relaksacija

(Meta)heuristike se primenjuju uglavnom za neke složene probleme gde egzaktni algoritmi ili ne postoje ili, ako postoje, mogu biti primenjeni samo na probleme malih dimenzija. Heuristike

su daleko razumljivije donosiocu odluke od klasičnih optimizacionih metoda, pa to može povećati šanse za njihovu primenu u praksi. Heuristike je značajno lakše realizovati na računaru. Heuristike se lako modifikuju kada dodje do nekih neočekivanih promena u realnom problemu. Mana heuristika je u tome što se ne garantuje pronalaženje optimalnih rešenja, teško ih je matematički analizirati i teško je odrediti pravila po kojima bi se one empirijski poredile. Metaheuristike pružaju moguće odgovore na teškoće koje mogu nastati pri rešavanju nekog problema, kao što su kako izaći iz lokalnog optimuma, kako nastaviti pretraživanje u oblasti dopustivog skupa koji ranije nije dovoljno istražen (problem diversifikacije), kako fokusirati pažnju na atraktivne oblasti dopustivog skupa (problem intensifikacije) i kako iskoristiti dobra rešenja dobijena u toku pretrage razbacana po prostoru rešenja? Odnosno da li se nekim kombinovanjem tih dobrih rešenja može dobiti neko još bolje rešenje [203].

2.3.1 Egzaktni algoritmi

Grananje sa ogradjivanjem (engl. Branch and bound) je tehnika koja je definisana kao razbijanje jednog problema kombinatorne optimizacije na više manjih problema. U toku postupka rešavaju se relaksacije problema. Grananje sa odsecanjem (engl. Branch and cut) se definiše na način da postoji ograničenje da neke od promenljivih moraju uzimati vrednost iz skupa celih brojeva (ili nekog njegovog podskupa). Metoda koristi niz linearnih relaksacija početnog problema. Metoda odsecanja (engl. Cutting plane method) odseca necelobrojno nezadovoljavajuće rešenje relaksiranog problema.
U ovom odeljku detaljnije se opisuje metoda Grananje sa ogradjivanjem koja se koristi u istraživanju ove Disertacije. Rezultati primene ove metode su prikazani u Poglavlju 5, odeljku Metode promenljivih formulacija i stepena čvora višeg reda u rešavanju problema maksimuma klike. U Poglavlju 3, u odeljku Pregled radova iz oblasti MC dat je prikaz radova koji rešavaju maksimalnu kliku na osnovu ove metode. Ona se do sada najčešće koristila u testiranjima pomenutog problema u odnosu na ostale egzaktne metode, stoga se može smatrati najzastupljenijom.

Metoda grananja i ogradjivanja

Metoda grananja i ogradjivanja se bazira na principu "podeli pa vladaj". Prvi su je predložili Land A.H. i Doig A.G. 1960 godine [155]. Kako je početni problem suviše složen da bi bio direktno rešen, deli se na manje i manje delove koji se mogu rešiti. Deljenje prostora rešenja obavlja se ogradjivanjem vrednosti pojedinih varijabli, dok se procenom maksimalne vrednosti rešenja u pojedinom delu mogu odbaciti delovi prostora rešenja koja ne daju najbolje vrednosti funkcije cilja. Loša osobina metode je nepolinomijalnost, dok je dobra osobina što se u praksi ova metoda završava u dopustivoj tački koja je blizu rešenja. Do trenutno najboljeg rešenja se može doći neposrednom proverom ili nekim heurističkim postupcima u toku algoritma. Metoda grananja i ogradjivanja sastoji se od tri osnovna koraka: grananje (engl. Branch), ogradjivanja (engl. Bound) i procenjivanje (engl. Fathoming). Posmatrajmo problem:

$$\min f(x), x \in X \tag{2.6}$$

gde je X diskretan skup. Problem 2.6 se razlaže na potprobleme sa ciljem da se neki od potproblema neće ni rešavati ako se ustanovi da nemaju bolje rešenje od trenutno najboljeg rešenja. Grananje problema 2.6 na potprobleme se može vršiti tako što se skup X pokriva unijom svojih delova $X_k, k \in K$ i problem 2.6 zameni skupom potproblema:

$$\min f(x), x \in X_k \tag{2.7}$$

Optimalno rešenje polaznog problema je optimalno za bar jedan problem sa spiska. Razlaganje skupa X na podskupove je pogodno zato što skupovi mogu biti dovoljno sitni ili laki za rešavanje, pa se vrednost funkcije na njima može lakše oceniti. U slučaju da je potproblem težak za rešavanje, olakša se izostavljanjem nekih ograničenja (umesto funkcije $f(x)$ posmatra se funkcija $g(x)$ u kojoj je neko ograničenje izostavljeno) ili proširivanjem njegovog dopustivog skupa X_k na neki skup Y_k. Na taj način se dobija problem:

$$\min g(x), x \in Y_k \tag{2.8}$$

koji se naziva relaksirani problem. Optimalna vrednost relaksiranog problema Q_k nije veća od optimalne vrednosti problema P_k, tj. važi $v(Q_k) \leq v(P_k)$.

U idealnom slučaju optimalne vrednosti su jednake i početni problem je rešen. Ako problem nije rešen, dobija se ocena odozdo $m(P_k) = v(Q_k)$ njegove optimalne vrednosti. Neka je, npr. trenutno najbolje pronadjeno rešenje M ili je najbolje rešenje ograničeno sa M:

$$m(P_k) \geq M$$

U skupu X_k nema rešenja koje je bolje od onog koje je dobijeno kao tekući rekord zato što je:

$$v(P_k) \geq m(P_k) \geq M \geq v(P)$$

U trenutku kada se proceni da problem neće dati bolja rešenja od tekućih, prelazi se na sledeći potproblem, a tekući problem se eliminiše sa spiska potproblema. U svakom grananju generiše se samo konačan broj potproblema, i ako se oni nikada ne ponavljaju, algoritam se posle konačno mnogo iteracija završava nalaženjem optimalnih vrednosti i rešenja polaznog problema. Ovaj postupak se najlakše može prikazati preko grafa čiji su čvorovi dobijeni grananjem. Dobijeni graf se naziva drvetom pretraživanja u kome je koren polazni problem a listovi najsitniji problemi (problemi koji ne mogu dati bolje rešenje daljim grananjem). Pri izboru problema za grananje, postoje dve osnovne taktike: Grananje u širinu, Grananje u dubinu.
U početnoj fazi poželjnije je koristiti grananje u dubinu, jer se na brži način dolazi do dopustivog rešenja, dok je u kasnijoj fazi bolja taktika grananja u širinu, jer daje bolja odsecanja.

2.3.2 Heuristike i metaheuristike

Sledeći pregled heurističkih metoda je važan sa stanovišta zastupljenosti i testiranja kod istraživača širom sveta u poslednjih nekoliko decenija. Sve navedene metode su osnov za poredjenje sa novijim radovima te se smatraju referentnim tačkama od kojih se polazi u većini istraživanja. U ovoj Disertaciji poredili smo se neposredno i posredno sa svim ovim heuristikama što je dato u Poglavlju 5.
U Poglavlju 3, odeljku Pregled radova iz oblasti MC dat je pregled radova koji su koristili prikazane heuristike prilikom rešavanja problema maksimalne klike.

Simulirano kaljenje

Proces kaljenja se koristi u fizici kako bi se dobila čista rešetkasta struktura. Proces se sastoji od topljenja krute materije zagrevanjem a kasnije stvrdnjavanja preko laganog hladjenja do stanja niske energije. Kao rezultat ovog procesa, slobodna energija sistema je minimizovana. Ta karakteristika procesa kaljenja se koristi za svrhu optimizacije, gde svako dopustivo rešenje sjedinjuje stanje hipotetičkog fizičkog sistema, a funkcija cilja reprezentuje energiju koja je u vezi sa stanjem. Bazična ideja simuliranog kaljenja je takva da se generiše dopustivo rešenje $x^{(0)}$, zatim se u iteraciji $k+1$, dobija da dopustivo rešenje $x^{(k)}$ prihvata suseda $x^{(k+1)}$ kao sledeće dopustivo rešenje sa verovatnoćom:

$$p_{k+1} = \begin{cases} 1 & \text{ako je } f(x^{k+1}) < f(x^k) \\ \exp\left(\dfrac{f(x^k) - f(x^{k+1})}{t}\right) & \text{inače} \end{cases}$$

Funkcija $f(x)$ je funkcija cilja a parametar t predstavlja temperaturu, koja se modifikuje kako se procedura optimizacije izvršava. Pravilan izbor rasporeda hladjenja opisujući promenu u temperaturnom parametru predstavlja najvažniji deo algoritma. Logaritamski spor raspored hladjenja dovodi do optimalnog rešenja u eksponencijalnom vremenu, dok u praksi, brži raspored hladjenja dovodi do prihvatljivih rešenja.

Neuronske mreže

Neuronske (neuralne) mreže predstavljaju poduhvat da se imitira neka korisna karakteristika biološkog nervnog sistema, kao što je adaptivno biološko učenje. Sastoje se od velikog broja visoko povezanih procesnih elemenata koji imitiraju vezane neurone sa težinskim relacijama -

analogonima sinapsama. Baš kao biološki nervni sistem, neuralne mreže se karakterišu masivnim paralelizmom i visokim unutrašnjim vezama. Iako su neuralne mreže bile predstavljene kasnih 50-tih godina prošlog veka, nisu se široko primenjivale do sredine 80-tih godina prošlog veka, kada se razvila dovoljno napredna metodologija. Hopfield i Tank [126] su dokazali da se odredjeni modeli neuralnih mreža mogu koristiti da se aproksimativno reše neki teški problemi kombinatorne optimizacije. Danas se neuralne mreže primenjuju na mnoge teške probleme iz stvarnog života. Detalji o neuralnim mrežama se mogu pronaći kod autora Zhanga [225].

Genetski algoritmi

Genetski algoritmi u optimizaciji su motivisani procesima evolucije u prirodnim sistemima. Optimizacija u genetskom algoritmu se izvodi na populaciji tačaka koje se zovu individue ili hromozomi. Najjednostavnije, hromozomi su predstavljeni binarnim vektorima. Svaki hromozom je povezan sa prilagodjenom vrednošću, odnosno sa verovatnoćom da će individua definisana preko hromozoma i u sledećoj generaciji preživeti. Funkcija prilagodjenosti individue se koristi da se analiziraju vrednosti funkcije cilja kako bi se odredio kvalitet jedinke. Osnovni zakon prirodne selekcije se zasniva na tome da je ukupna prilagodjena vrednost populacije ne opadajuća iz generacije u generaciju, i to se uzima kao baza optimizacione procedure. U najjednostavnijoj verziji, genetski algoritam počinje sa populacijom koja je slučajno izabrana i računa novu populaciju koristeći jedan od tri bazna mehanizma: selekciju, ukrštanje ili mutaciju [99]. Operator selekcije bira hromozome koji se koriste u sledećoj generaciji s obzirom na verovatnoću koja je data preko vrednosti odgovarajuće funkcije prilagodjenosti. Operator ukrštanja se primenjuje kako bi se dobila nova deca od parova individua. Konačno, operator mutacije menja vrednost svakog bita u hromozomu sa ciljem obnavljanja genetskog materijala izgubljenog ukrštanjem.

Pohlepna slučajna adaptivna procedura pretrage

Pohlepna slučajna adaptivna procedura pretrage (engl. Greedy randomized adaptive search procedure, GRASP) je iterativna metoda slučajnih uzoraka u kom svaka iteracija omogućava da se dodje do heurističkog rešenja problema [85]. Najbolje rešenje u svim GRASP iteracijama se čuva kao konačan rezultat. Postoje dve faze u okviru svake GRASP iteracije: prva konstruiše listu rešenja koja se naziva lista ograničenih kandidata (engl. Restricted candidate list, RCL) preko adaptivne slučajne pohlepne funkcije; druga primenjuje tehniku lokalne pretrage u konstruisanim rešenjima u nadi da će se naći poboljšanje.

Tabu pretraga

Tabu pretraga [94] je varijacija algoritama lokalne pretrage, koja koristi tabu strategiju kako bi se izbeglo ponavljanje tokom pretrage prostora dopustivih rešenja. Takva strategija je implementirana kreiranjem skupa tabu listi, koje sadrže putanje sastavljene od dopustivih rešenja do kojih se došlo u prethodnim algoritamskim koracima. Sledeće dopustivo rešenje se definiše kao najbolje legalno rešenje (nije zabranjeno tabu strategijom) u okolini trenutnog rešenja, čak iako je gore od trenutnog rešenja. Postupak pretrage se ne prekida u trenutku dolaska do lokalnog optimuma, već se pokušava izaći iz tog optimuma u potrazi za drugim rešenjem. U momentu napuštanja lokalnog optimuma uvodi se zabrana povratka u ranije posećena rešenja i to na taj način što se označavaju atributi problematičnih rešenja. Jedan od najbitnijih parametara u metodi je ograničenje odredjenog broja koraka kretanja kroz prostor rešenja. Atributi se čuvaju u listi koja se naziva tabu lista. Ovo je veoma poznata metoda na kojoj se radilo, tako da je kasnije upotpunjena novim metodama intensifikacije i diversifikacije. Intensifikacija je formiranje skladišta dobrih rešenja pronadjenih u toku pretrage (engl. Elite solutions). Diversifikacija podrazumeva da se pretraga rešenja nastavlja u nekom drugom delu prostora rešenja. Obično se vrši kada se zaluta u neki deo prostora rešenja gde ne može da se stigne do kvalitetnijeg rešenja.

Metoda promenljivih okolina

Metodi promenljivih okolina [110] detaljno je posvećeno četvrto poglavlje. VNS je osmišljena da koristi više od jedne strukture okolina i da ih sistematično menja u okviru lokalne pretrage. Algoritam ostaje u istom rešenju sve dok se drugo rešenje koje je bolje od trenutnog ne pronadje.

Tada se vrši pomeraj u to bolje rešenje. Okoline se najčešće rangiraju na taj način što je pretraga oko trenutnog rešenja intenfizikovana i diversifikovana. Nivo intenzifikacije i diversifikacije se može kontrolisati preko nekoliko parametara koji se lako podešavaju. Ceo VNS se može posmatrati kao proces razmrdavanja gde pokret ka okolini koja je dalje od trenutnog rešenja odgovara jačem razmrdavanju. VNS dozvoljava kontrolisan rast u okviru nivoa razmdravanja.

Heuristike bazirane na neprekidnoj optimizaciji

Softverski paket GLOB

Softverski paket GLOB je relevantan u ovom odeljku sa stanovišta da se koristio u istraživanju ove Disertacije. Intervenisali smo u okviru parametara i funkcija paketa i posmatrali kako se paket ponaša prilikom promena istih. Rezultati su dati u Poglavlju 5. GLOB je baziran na metodologiji VNS metaheuristike [76]. GLOB je programiran za probleme sa ograničenjima i bez ograničenja. Softver je osmišljen kao samostalna alatka za minimizaciju kontinualne funkcije sa i bez ograničenja. Kod je pisan u ANSI C programskom jeziku i sastoji se od oko 10000 linija. Veličina koda je bitna da se istakne zbog raznolikosti metoda koje su programirane i oblikovane u integralnu VNS strategiju. Paket se sastoji od 4 osnovne verzije metode: Osnovne metode promenljivih okolina, Multistart lokalne pretrage, Monte Karlo metode i Gausove metode promenljivih okolina. Moguće je dodati nove metode. Pored metoda, isprogramirano je dosta funkcija za lokalnu optimizaciju tako da su pokrivene različite familije funkcija, od totalno diferencijabilnih, preko diferencijabilnih u tački do nediferencijabilnih. Paket se sastoji od raznih funkcija za statističko izveštavanje koje služe istraživanju i komparaciji rešenja. GLOB je moguće prilagoditi da bude komercijalan softver, pošto je u osnovi zadovoljio sve kriterijume izrade i implementacije softvera, ali je njegov osnovni doprinos u razumevanju heuristika prilikom rešavanja problema globalne optimizacije.

Pregled GLOB alata

GLOB je osnovno dizajniran da pretraži globalni minimum kontinualne (glatke) funkcije na konačno dimenzionom regionu koji je boks ograničen.

$$\min f(x_1, x_2, ..., x_n), a_i \leq x_i \leq b_i, i = 1, .., n$$

Broj promenljivih (dimenzija problema) je postavljen na maksimum 200 ali je to moguće menjati. Korisnik definiše funkciju f i njen gradient kao:

$$double \quad user_{function}(double \quad * xcoordinates)$$

i

$$void \quad user_{gradient}(double \quad * xcoordinates, double \quad * gradient)$$

Ukoliko funkcija koja se minimizuje zavisi od nekih parametara sa fiksnim vrednostima u procesu minimizacije, kao kod nekih testnih funkcija, njihove vrednosti se mogu definisati pomoću fun_{params} opcije u fajlu koji definiše vrednosti u vektoru $Fun_{Params}[]$ koje se koriste u korisničkoj funkciji. Postoje brojne dobro poznate test funkcije koje su već kreirane u paketu. To su Rosenbrock, Shekel, Hartman, Rastrigin, Shubert, Branin, Baluja i druge, a takodje moguće je kreirati nove, pa je funkcija maksimalne klike u kontinualnoj formulaciji isprogramirana. Kako je paket programiran da radi u batch modu, svi parametri su definisani u okviru glavnog fajla parametara i job fajla parametara. U glavnom fajlu parametara (glob.cfg) korisnik specificira koji job fajl parametara će se koristiti, ime izlaznog fajla i radni direktorijum. Job fajl parametara sadrži testne funkcije, različita ograničenja i opcije koje podešavaju izveštavanje za taj job. Ovaj pristup omogućava da se svaki optimizacijski job u batch modu prihvati.

3

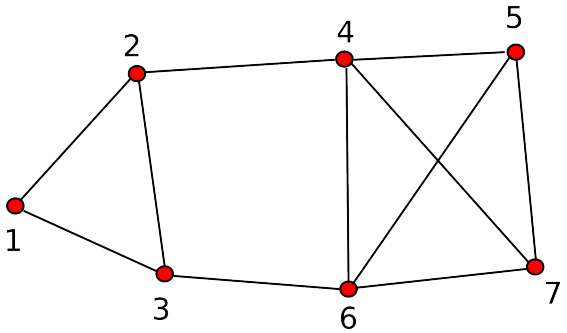
Osnove teorije grafova i formulacije problema maksimalne klike

Teorija grafova izučava matematičke strukture koje se zovu grafovi. **Graf** se prikazuje uredjenim parom $G = (V, E)$. V je konačan, neprazan skup čvorova (vrhova, temena), a E je jedan skup dvoelementnih podskupova skupa V, odnosno skup ivica (lukova, grana). Engleski termini za čvorove su node ili vertex. Za ivice se koriste termini edge ili link. Grafovi se dele na **neorijentisane i orijentisane**, zavisno od toga da li je ivica koja spaja čvorove u i v isto što i ivica koja spaja čvorove v i u [118].

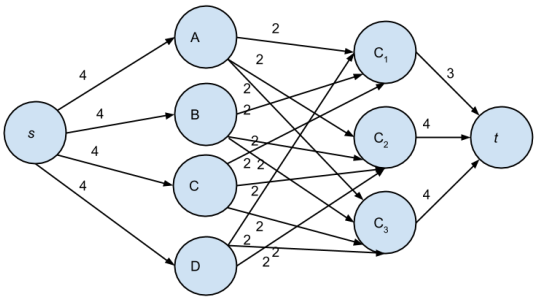
3.1 Definicije i notacije

Susedi su oni čvorovi izmedju kojih postoji ivica. Dva čvora su susedna ukoliko postoji ivica koja ih spaja, odnosno $u, v \in V$ su susedni ako $e = \{u, v\} \in E$. Neka je $G = (V, E)$ neorijentisani graf i $v \in V$ čvor grafa G, onda je $N(v)$ skup svih suseda čvora v je $N(v) = \{u \in V | (v, u) \in E\}$. **Stepen** (degree) čvora v se označava sa $d(v)$ i predstavlja broj suseda čvora v. **Petlja** je ivica koja spaja čvor sa samim sobom. Graf koji nema petlje niti paralelne ivice se naziva **prost**.

Dva najjednostavnija primera grafova su **kompletan (potpuni)** graf K_n koji predstavlja prost graf u kom su svaka dva čvora susedna [24] i **nula** graf koji predstavlja graf izolovanih čvorova, tj. graf koji nema ivica. Kompletan k-partitan graf jeste graf koji se može podeliti u k stabilnih skupova tako da je svaki par čvorova iz dva različita stabilna skupa povezan.
Dve ivice u grafu su **incidentne** ako imaju jednu krajnju tačku zajedničku. **Put** dužine $k, k \geq 1$ u grafu (V, E) je niz ivica iz E oblika $(v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k)$ kod orijentisanih grafova i $\{v_0, v_1\}, \{v_1, v_2\}, ..., \{v_{k-1}, v_k\}$ kod neorijentisanih grafova. Put je **prost** ako se nijedan čvor ne pojavljuje više puta u nizu. Graf je povezan ako su svaka dva njegova čvora povezana, odnosno ako izmedju svaka dva čvora postoji put. Kada se ivicama grafa pridruže brojevi (dužine, cene, težine, pouzdanost, propusna moć), takvi grafovi se nazivaju **težinski ili obojeni**. Pridruženi brojevi su najčešće pozitivni, mada i ne moraju. Moguće je da se težine dodele i čvorovima



Slika 3.1: Primer neorijentisanog grafa



Slika 3.2: Primer težinskog grafa

grafa a ne samo ivicama. U engleskom jeziku se za grafove u kojima su težine dodeljene ivicama koristi termin **edge–weighted**, a za grafove u kojima su težine dodeljene čvorovima termin **node–weighted** grafovi [216].

Predstavljanje grafa na računaru

Postoje različite varijante predstavljanja grafa na računaru, a svaka od njih zavisi od prirode problema koji se rešava i računarskih resursa kojima raspolaže. Pod pojmom računarski resursi, uglavnom se misli na raspoloživi memorijski prostor. Obično se čvorovi grafa numerišu brojevima $0, 1, 2, ..., n-1$ (ili $1, 2, ..., n$) gde je n broj čvorova u grafu. Skup ivica se reprezentuje na jedan od dva načina:

Matrica susedstva predstavlja elemente koji daju informaciju o tome da li postoje ivice izmedju čvorova koji odgovaraju indeksima tih elemenata. Ako graf nije težinski onda elementi matrice susedstva samo sadrže informaciju o postojanju ivice izmedju odgovarajućih čvorova. Ako je graf težinski, onda element matrice sadrži informaciju o težini ivica.

Formalno, matrica susedstva dimenzije $n \times n$ od G se zapisuje kao

$$A_G = (a_{ij})_{(i,j) \in V \times V}$$

gde je $a_{i,j} = 1$ ako $(i, j) \in E$ (jeste ivica u G), a $a_{i,j} = 0$ ako $(i, j) \notin E$ [190].

Matrica susedstva je simetrična matrica. Na primeru grafa sa slike 3.1 dobija se da je matrica susedstva:

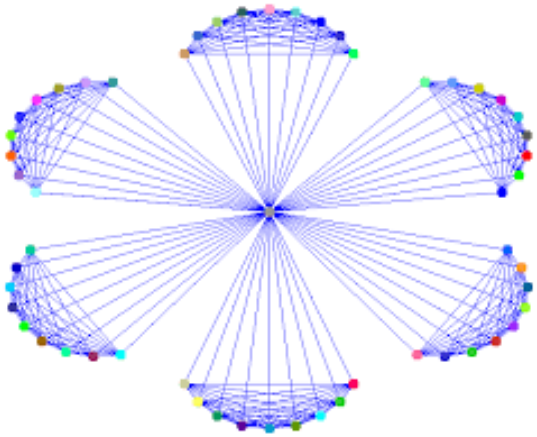
$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Definiše se spektralni radius matrice susedstva G, $\rho(A_G)$, kao najveća sopstvena vrednost od A_G. Karakteristične (sopstvene) vrednosti A_G su $\lambda_1, ..., \lambda_n$, gde su λ_i skalari pridruženi karakterističnom (sopstvenom) vektoru. Karakterističan (sopstveni) vektor predstavlja nenula vektor x koji se linearnom transformacijom transformiše u sebi kolinearan vektor:

$$Ax = \lambda x$$

Graf se moze predstaviti **listama suseda** tako što je svaki od čvorova grafa element prema kome se formira lista u kojoj su smešteni susedi tog čvora u grafu [157]. Kada se radi sa retkim grafom (graf u kojem je broj ivica proporcionalan broju čvorova), koriste se liste suseda jer se sve ivice grafa opisuju sa relativno malo podataka.

Ipak, varijanta sa listom je skupa jer na primer jedna vrlo jednostavna operacija kao što je provera da li su neka dva čvora susedi, zahteva pregledanje liste suseda. U praksi, broj takvih provera može biti vrlo veliki i performanse programa će biti značajno pokvarene. Ukoliko se raspolaže sa dovoljno memorije, onda je najbolje predstaviti graf na oba načina, da bi se nakon toga koristile informacije ili iz matrice susedstva ili iz liste suseda, zavisno od toga gde se nalaze informacije koje su bitne. Lista suseda se obično radi na osnovu rastojanja, tako da bude prvi u



Slika 3.3: Primer gustog grafa

listi čvor–sused koji je najbliži, a poslednji čvor–sused koji je najudaljeniji [118].
Neka je $G = (V, E)$ prost neorijentisan graf sa skupom čvorova $V = \{v_1, v_2, ..., v_n\}$ i sa skupom ivica $E = \{e_1, e_2, ..., e_m\}$. **Komplement** grafa G je graf $\bar{G} = (V, \bar{E})$, gde je $\bar{E}$ komplement od E. Za podskup W u oznaci $W \subseteq V$, važi da je $G(W)$ podgraf indukovan sa W na G. $N(i)$ je **skup suseda** čvora i i $d_i = |N(i)|$ je stepen čvora i. Sa $\delta = \delta(G)$ označava se **maksimum stepena** čvorova iz G.

3.2 Kombinatorni problemi na grafovima

Svedoci smo široke upotrebe grafova u realnom životu, primena grafova na strukture koje se izgrađuju navodi na to da se problemi koji se u toku poslovanja generišu rešavaju uz pomoć algoritama razvijenih za grafove. Primene grafova su široke: internet, telekomunikacione mreže, putna mreža u jednoj (ili više) zemalja, železnička mreža, poštanski saobraćaj, sistem za prenos električne energije, štampane ploče itd [74]. Neki od problema koji se uspešno rešavaju su optimalni transport robe, optimalni prenos električne energije, zaštita od havarija, obezbedjivanje uslova da postoje alternativni putevi za prenos energije u slučaju otkaza itd. Jedna od prvih poteškoća je ta što mnogi od navedenih problema spadaju u klasu NP-kompletnih (NP-teških problema), pošto je za takve probleme nalaženje optimalnog rešenja skoro neizvodljivo u realnom vremenu (vidi Poglavlje 3).

Rešavanje NP teških problema egzaktnim metodama je skupo i često nemoguće uraditi, stoga se najčešće pronalazi neka heuristika pomoću koje se dobija približno rešenje [118]. To rešenje ne mora biti optimalno, ali ako je heuristika dovoljno dobra i rešenje će biti blizu optimalnog. Problemi koji su vezani za grafove pripadaju problemima kombinatorne optimizacije. Za većinu njih je moguće zapisati formulaciju u duhu matematičkog programiranja. Za pojedine od problema postoje algoritmi sa polinomskim vremenom složenosti. Ali postoji i klasa problema za koje je pokazano da su NP-kompletni [157]. Sledi nekoliko primera optimizacionih problema na grafovima za koje je karakteristično da jesu NP–kompletni problemi.

Problem orijentisanog Hamiltonovog puta (engl. Directed hamiltonian path): Dati su orijentisan graf G i dva čvora u i v, koji pripadaju skupu čvorova toga grafa. Zadatak je da se utvrdi da li postoji put od čvora u do čvora v koji prolazi kroz svaki čvor grafa G tačno jedanput.
Problem Hamiltonove petlje (engl. Directed hamiltonian circuit): Dat je orijentisan graf G. Zadatak je da se utvrdi da li u grafu G postoji Hamiltonova petlja. Hamiltonova petlja je put kojem se poklapaju početak i kraj i koji prolazi kroz svaki čvor grafa (osim kroz početni, odnosno krajnji) tačno jedanput.
Problem k–bojenja grafa (engl. K-colouring problem): Dati su graf G i ceo broj k ($1 \leq k \leq n = |G|$). Zadatak je da se utvrdi da li postoji korektno bojenje čvorova grafa sa ne više od k različitih boja. Neko bojenje čvorova grafa je korektno, ako ne postoji niti jedna ivica e tog grafa

čiji su krajevi obojeni istom bojom. Pokazuje se da je za $k = 3$ (a kasnije i za $k > 3$) problem bojenja NP-kompletan. Postoji takodje problem bojenja ivica, što znači da svake dve ivice sa jednim zajedničkim krajem imaju različitu boju. Pokazuje se da je problem 3–bojenja ivica grafa isto tako NP–kompletan problem [157]. Problem 3–bojenja ivica je NP–kompletan čak i u slučaju kada je graf koji se boji 3–regularan (tj. svaki čvor tog grafa ima stepen 3). Minimalni k za koji G prati k-bojenje se naziva **hromatski broj** i označava se sa $\chi(G)$. Problem bojenja grafa je pronaći $\chi(G)$ i particiju čvorova indukovanu sa $\chi(G)$ bojenja.
Problem minimalne particije klike koji se bavi pitanjem kako podeliti čvorove tako da se dobiju minimalni brojevi klika je analogan problemu bojenja grafova. Bilo koje pravilno bojenje u G je particija klike u $\bar{G}$. Ostale standardne definicije koje se koriste u radu i koje su u vezi sa razmatranim problemom se nalaze u datim referencama o teoriji grafova [74].
Problem dominirajućeg skupa (engl. Minimum dominating set): Neki podskup $V' \subset V$ skupa čvorova V grafa $G = (V, E)$ je dominirajući ako i samo ako je svaki čvor $v \in V \setminus V'$ povezan ivicom sa bar jednim čvorom iz V'. Neka je dat graf $G = (V, E)$ i ceo broj $k \leq n = |V|$. Zadatak je da se utvdi da li postoji dominirajući skup sa najviše k elemenata [74].

Problem najdužeg puta (engl. Longest path problem): Neka su dati graf $G = (V, E)$, dva izdvojena čvora u i v iz skupa V i ceo broj $k \leq n = |V|$. Zadatak je da se utvrdi da li postoji prost put od čvora u do čvora v čija dužina nije manja od k. Pod dužinom se podrazumeva broj ivica na tom putu.

Problem izomorfnog podgrafa (engl. Subgraph isomorphism): Grafovi $G_1 = (V_1, E_1)$ i $G_2 = (V_2, E_2)$ su izomorfni ako i samo ako postoji bijektivno preslikavanje $f : V_1 \to V_2$ skupa čvorova grafa u skup čvorova drugog grafa tako da za proizvoljna dva čvora v_1 i v_2 grafa G_1 važi da postoji ivica izmedju čvorova v_1 i v_2 ako i samo ako postoji ivica izmedju $f(v_1)$ i $f(v_2)$. Problem izomorfnog podgrafa se može formulisati i na sledeći način: Data su dva neorijentisana grafa G_1 i G_2. Utvrditi da li postoji podgraf G_3 grafa G_1 koji je izomorfan grafu G_2.

3.3 Maksimalna klika

Odeljak detaljnije objašnjava maksimalnu kliku i maksimum klike.
Skup $C \subseteq V$ čvorova je **klika** ako je svaki od čvorova susedan sa svima ostalima (ako i samo ako su svaka dva čvora iz skupa C povezana). Klika C je podskup od V tako da je podgraf $G(C)$ indukovan sa C na G kompletan. Problem maksimalne klike se može formulisati na sledeći način:
U datom neorijentisanom grafu odrediti kliku koja ima najviše elemenata, odnosno problem maksimalne klike je problem u kom se pronalazi klika maksimalne kardinalnosti.
Maksimalna klika u grafu je klika sa maksimalnim brojem čvorova. [28].
Maksimalna klika se ne sadrži ni u jednoj drugoj kliki u G.
Kao što je napomenuto, pored maksimalne klike razmatraju se i problemi maksimalnog stabilnog skupa i minimalnog čvornog pokrivača, s obzirom da su pomenuti problemi komplementarni i povezani. Na osnovu toga, date su još neke definicije pomenutih problema.
Skup $S \subseteq V$ čvorova je **stabilan** ili nezavisan, ako su bilo koja dva čvora u skupu nesusedna, odnosno ako je skup ivica podgrafa indukovanog sa S prazan. Stabilni skup je maksimalan ako nije podskup bilo kog većeg stabilnog skupa i maksimum je ako ne postoji veći stabilni skup u grafu.
Stabilni skup maksimalne kardinalnosti (engl. Stable set of maximum cardinality) u grafu G jeste stabilan skup najveće veličine.
Problem odredjivanja skupa čvorova koji predstavljaju maksimalni stabilni skup (engl. Stable–set problem)se definiše na sledeći način: Ako je dat neorijentisani graf $G = (V, E)$, odrediti stabilni skup V' tog grafa najveće moguće veličine [157].
Skup $T \subseteq V$ čvorova je **čvorni pokrivač** ili transverzalan ako bilo koja ivica u E sadrži najmanje jedan čvor u T. Minimum čvornog pokrivača (engl. Minimum vertex cover) jeste najmanji skup čvorova takvih da je svaka ivica u grafu G incidentna sa najmanje jednim čvorom u skupu.
Problem odredjivanja skupa čvorova koji predstavlja pokrivač (engl. Vertex–cover problem)se definiše na sledeći način: Ako je dat neorijentisani graf $G = (V, E)$ i broj k $(1 \leq k \leq n)$,

utvrditi da li postoji pokrivač V' tog grafa koji ima najviše k čvorova [74].
Takodje, formulacija problema pokrivača je navedena u duhu matematičkog programiranja [157]. Ako graf G ima n čvorova, onda je te čvorove moguće numerisati brojevima od 1 do n. Svaki pokrivač je definisan karakterističnim vektorom x koji ima n elemenata.
Stoga je $x_i = 1$ ako čvor i pripada pokrivaču i $x_i = 0$ ako čvor i nije u pokrivaču. Tako formulacija ima sledeći zapis

$$\min \sum_{i=1}^{n} x_i$$

tako da je

$$x_i + x_j \geq 1, \forall e = (i,j) \in E, x_i \in \{0,1\}$$

Kao što se vidi, to postaje problem 0-1 kombinatorne optimizacije. Broj ograničenja je jednak broju ivica tog grafa te, u opštem slučaju, može biti ukupno $O(n^2)$ ograničenja.

Klika je maksimalna ako se ne može dodati niti jedan novi čvor tako da i dalje ostane klika.
Maksimum klike C od G je klika maksimalne veličine [190].
Treba odvojiti maksimum klike (engl. Maximum clique, MC) od maksimalne klike (engl. Maximal clique) vidi sliku 3.1. Maksimalna klika (stabilni skup) je klika (stabilni skup) koja nije pravi podskup bilo koje druge klike (stabilnog skupa). Maksimum klike (stabilnog skupa) je maksimalna klika (stabilni skup) koja ima najveću kardinalnost. Na primeru datog grafa sa slike 3.1 vidi se da je maksimum klike podgraf koga čine čvorovi $\{4,5,6,7\}$, a maksimalne klike su date podgrafovima $\{1,2,3\}$ i $\{4,5,6,7\}$.

Sa $\omega(G)$ se označava **broj elemenata** u kliki koja ima najviše elemenata. U literaturi na engleskom jeziku taj broj se naziva **broj klike (red klike)** (engl. Clique number). Slično se sa $\alpha(G)$ označava broj elemenata u stabilnom skupu koji ima najviše elemenata i taj broj se naziva **stabilan broj** (engl. Stability number). Drugim rečima, broj klike $\omega(G) = \max\{|C|\}$, C je klika od G jednak je kardinalnosti maksimalne klike, odnosno, to je broj čvorova u maksimalnoj kliki u G.
Takođe, **stabilan broj skupa (red stabilnog skupa)**, $\alpha(G)$, je kardinalnost maksimalnog stabilnog skupa u G.
Pokazalo se da je MC problem ekvivalentan sa problemom stabilnog skupa, kao i sa problemom minimalnog čvornog pokrivača, a bilo koji algoritam za MC se može direktno primeniti na ove važne probleme. C je maksimalna klika na G akko je S stabilni skup na $\bar{G}$ i akko je $V \setminus S$ minimalni čvorni pokrivač na G. Činjenica proističe iz identiteta Gallai-a [88]:

$$\alpha(G) + |T| = |V(G)|$$

gde je T minimum čvornog pokrivača u grafu G. U odnosu na blisku relaciju izmedju maksimalnog stabilnog skupa i problema maksimalne klike, vrši se paralelna analiza tih problema sa objašnjavanjem i opisivanjem karakteristika i algoritama za problem maksimalne klike. Jasno je da je svaki rezultat vezan za problem maksimalne klike u G, vezan i za problem stabilnog skupa u $\bar{G}$. U slučaju težinskog grafa težina $W(X)$ podskupa $X \subseteq V$ je definisana:

$$W(X) = \sum_{i \in X} w_i$$

Kao što se može videti, maksimum klike se odnosi na sume težina [190]. Problem maksimalne težinske klike traga za klikom maksimalne težine. Problem maksimalnog težinskog stabilnog skupa traga za stabilnim skupom maksimalne težine.
U nekim aplikacijama, pored klika nailazi se i na guste podgrafove ili kvazi klike.
Kvazi klika C_γ, predstavlja podskup čvorova V tako da $G(C_\gamma)$ ima najmanje $\lfloor \gamma q(q-1)/2 \rfloor$ ivica, gde je q kardinalnost od C_γ i γ je koeficijent koji se podešava testiranjem tako da važi da $q\gamma$ bude maksimalno.

Donja i gornja granica maksimalne kardinalnosti klike

U ovom odeljku sažeto su predstavljene dobro poznate donje i gornje granice za broj klike i stabilan broj i diskutuje se kompleksnost izračunavanja.

U teoriji redova, gornja granica podskupa S nekog parcijalno uredjenog skupa $(K, \leq)$ jeste element od K i ona je veća ili jednaka svakom elementu u S [128]. Termin donja granica se definiše dualno kao element od K koji je manji ili jednak svakom elementu od S.
Za skup sa gornjom granicom se kaže da je ograničen sa gornje strane tom granicom, a za skup sa donjom granicom se kaže da je ograničen sa donje strane tom granicom. Termini ograničen sa gornje i ograničen sa donje strane se koriste u matematičkoj literaturi za skupove koji imaju gornju i donju granicu.
Najpoznatija donja granica se bazira na stepenu čvorova i data je u [50]:

$$\alpha(G) \geq \sum \frac{1}{d_i + 1} \tag{3.1}$$

Detaljnije, mnogi algoritmi koji računaju granice su bazirani na karakteristici matrice A_G. Kako neorijentisan graf može biti podeljen na povezane podgrafove, razmatraju se samo povezani grafovi. Oni imaju matrice susedstva sa karakteristikom da su neskrative. Neka su m broj ivica grafa i $\rho = \frac{2m}{n^2}$ je gustina grafa u matrici susedstva; za povezane grafove važi $2\frac{n-1}{n^2} \leq \rho \leq \frac{n-1}{n}$.

Jedna jednostavna granica se dobija iz [11] i glasi:

$$\omega(G) \leq \frac{3 + \sqrt{9 - 8(n-m)}}{2} \tag{3.2}$$

Najčešće citirana gornja granica koju je predložio Wilf [222] prvi put 1967. jeste

$$\omega(G) \leq \rho(A_G) + 1 \tag{3.3}$$

jednakost važi akko je graf kompletan. Kako bi se ova relacija dokazala koristi se da ako je x^C karakterističan vektor maksimuma klike, onda je

$$(x^C)^T A_G x^C = 1 - \frac{1}{\omega(G)}$$

i takodje

$$(x^C)^T x^C = \frac{1}{\omega(G)}$$

Granica se odredjuje preko generalne karakteristike $\frac{x^T A_G x}{x^T x} \leq \rho(A_G)$.

Wilf je uzeo u obzir karakterističan vektor x_P takav da je njegova karakteristična vrednost najveća a sve komponente vektora su pozitivne. On je normalizovan uz pomoć $s = e^T x_P$, gde je $\frac{x_P{}^T}{s} A_G \frac{x_P}{s} = \frac{\lambda_P}{s^2} \leq 1 - \frac{1}{\omega(G)}$. Na osnovu svega sledi:

$$\omega(G) \geq \frac{\lambda_P}{s^2 - \lambda_P} + 1 \geq \frac{\lambda_P}{n - \lambda_P} + 1 \tag{3.4}$$

sa važećom jednakosti akko je graf kompletan. Desna nejednakost se lako dobija iz $x_P{}^T x_P = 1$ i Cauchy-Schwarz nejednakost daje

$$s^2 = (e^T x_P)^2 \leq (e^T e)(x_P{}^T x_P) = n$$

Ukoliko se dobije potpun skup karakterističnih vektora i karakterističnih vrednosti od A_G ova granica se može poboljšati striktno. Sa potpunim skupom karakterističnih vektora može se napraviti $x^* \in \Delta$, tako da važi $g_* = (x^*)^T A_G x^* > \frac{\lambda_P}{s^2}$ [40] odakle se dobija

$$\omega(G) \geq \frac{1}{1 - g_*} \tag{3.5}$$

Neka je N_{-1} broj sopstvenih vrednosti A_G koje ne prekoračuju -1 i N_0 broj sopstvenih vrednosti jednakih 0. Amin i Hakimi [11] su dokazali

$$\omega(G) \leq N_{-1} + 1 < n - N_0 + 1 \tag{3.6}$$

s tim da važi jednakost ukoliko je graf kompletan multipartitan. Granica se može izračunati u $\mathcal{O}(n^3)$.
Jedna od vrlo dobrih metoda za računanje gornje granice stabilnog broja se bazira na semidefinitnom programiranju. Lovasz je 1979. prikazao teta broj (theta number) $\theta(G)$ koji daje gornju granicu stabilnog broja $\alpha(G)$ [159]. Ovaj broj je definisan kao optimalna vrednost sledećeg semidefinitnog programa:

$$\max e^T X e$$

p.o. $\mathrm{tr}(X) = 1$, $X_{ij} = 0$ ako $(i,j) \in E$ i $X \succeq 0$, X je simetrična matrica dimenzija $n \times n$, ograničenje $X \succeq 0$ zahteva da je X pozitivno semidefinitno, dok je e jedinični vektor dužine n. Poznata Sendvič teorema [150] dokazuje da je Lovasz-ov broj $\theta(\bar{G})$ od komplementa grafa izmedju broja klike $\omega(G)$ i hromatskog broja $\chi(G)$:

$$\omega(G) \leq \theta(\bar{G}) \leq \chi(G)$$

s obzirom da se semidefinitni program može rešiti u polinomijalnom vremenu, Sendvič teorema takodje pokazuje da se u savršenom grafu G (za koji važi $\omega(G) = \chi(G)$), broj klike može izračunati u polinomijalnom vremenu.

Geometrijska formulacija problema maksimuma klike [41] u kompleksnom prostoru produkuje granicu

$$\omega(G) \leq \frac{n + \bar{N}_0}{2} \tag{3.7}$$

gde je $\bar{N}_0$ broj sopstvenih vrednosti jednakih nuli matrice susedstva komplementa grafa $\bar{G}$. U ovom slučaju kalkulacija $\bar{N}_0$ se može obaviti u $\mathcal{O}(n^3)$. Dok se granica 3.2 može izračunati u $\mathcal{O}(n)$, granice 3.3, 3.6 i 3.7 zahtevaju više rada ali su obično stabilnije. Eksperimentalno, 3.6 je najstroža granica, mada se ništa ne može generalno zaključiti. Uvek se može pronaći graf za kog važi da su 3.3 i 3.7 najbolje granice. Otuda proizlazi da je najsigurnija strategija da se izračunaju sve granice i da se najstrožija izabere.
Kako bi se pronašlo g_* 3.5 potrebno je naći kompletan skup karakterističnih vrednosti i karakterističnih vektora od A_G i posledično ovu granicu je najteže izračunati, mada je ona najstroža. Poslednja očigledna tvrdnja je da iako $\omega(G)$ jeste celobrojno bilo koja ne celobrojna granica se može odrediti primenjujući operatore donjeg ili gornjeg zaokruživanja. Postoje slučajevi u kojima ove granice skoro mogu da reše problem maksimuma klike, kao na primer, za 64-čvorni graf "hamming6-2" sa DIMACS-a [144]. Primene 3.3 i 3.5 dozvoljavaju da se postave granice za graf $32 \leq \omega(G) \leq 33$. Uzimajući u obzir podskup svih mogućih grafova mogu se istražiti karakteristike datog podskupa kako bi se dobile strožije granice.

3.4 Diskretne formulacije problema maksimalne klike

U okviru odeljka Maksimalna klika, dali smo osnovnu definiciju pojma, dok u ovom odeljku pojam formalizujemo i proširujemo raznim poznatim formulacijama.
Formulacije maksimalne klike pomoću matematičkog programiranja

Jedna od najjednostavnijih formulacija problema maksimalne klike jeste formulacija preko grana grafa, tzv. edge formulacija.

Definicija 3.4.1 *Neka je dat skup V čvorova i skup E ivica u grafu G, onda je*

$$\max \sum_{i=1}^{n} x_i, \quad p.o. \quad x_i + x_j \leq 1, \forall (i,j) \in \bar{E}, \quad x_i \in \{0,1\}, \quad i = 1,..,n \tag{3.8}$$

gde je $x_i = 1$ ako čvor v_i pripada kliki, a 0 je inače.

Poliedarski rezultat koji je u vezi sa ovom formulacijom je usledio nakon dokaza koji su izveli Nemhauser i Trotter [180]. Oni su 1975. godine dokazali da ako je varijabla x_i celobrojna sa vrednošću 1 u optimalnom rešenju do na linearnu relaksaciju 3.8, onda je $x_i = 1$ u bar jednom optimalnom rešenju u 3.8.

Teorema 3.4.1 *Neka je x optimalno rešenje do na linearnu relaksaciju od 3.8, sa vrednostima $(0, 1/2, 1)$ i neka je $P = \{j|x_j = 1\}$. Tada postoji optimalno rešenje x^* u odnosu na 3.8, tako da je ${x_j}^* = 1, \forall j \in P$.*

Ova teorema predlaže implicitni algoritam nabrajanja za 3.8 putem rešavanja njegove linearne relaksacije problema. Nekoliko promenljivih imaju celobrojne vrednosti za optimalno rešenje u odnosu na linearnu relaksaciju 3.8. Razlika izmedju optimalne vrednosti u 3.8 i njegove linearne relaksacije problema je dosta velika što ozbiljno ograničava korišćenje ovog pristupa.
Sledi kvadratna $0-1$ formulacija [28]:
Neka je $I = n \times n$ jedinična matrica. Definiše se transformacija $t : \{0,1\}^n \to 2^V$, kao, $t(x) = \{i|x_i = 1, i \in V\}$, $\forall x \in \{0,1\}^n$. Drugim rečima t preslikava vektor na podskup kojim je taj vektor definisan, preslikavanje t preslikava skup vektora dimenzije n sa elementima 0 i 1 na partitivni skup skupa V na način da se svakom vektoru dodeljuje jedan podskup skupa V kojim je taj vektor definisan. Inverzno preslikavanje od t se označava sa t^{-1}. Neka je x^D karakterističan vektor definisan sa ${x_i}^D = 1/|D|$, ako je $i \in D$ i ${x_i}^D = 0$, inače.
Ako je $x = t^{-1}(D)$ za neki $D \in 2^V$, onda $x_i = 1$ ako $i \in D$ i $x_i = 0$ ako $i \notin D$, za $i = 1, .., n$, odnosno neka je $x = |D|x^D$.
Problem maksimizacije se može zapisati kao problem minimizacije kada je $x_i = 1$ onda

$$\min f(x) = -\sum_{i=1}^{n} x_i, \quad \text{p.o.} x_i + x_j \leq 1, \forall (i,j) \in \bar{E}, x \in \{0,1\}^n. \tag{3.9}$$

Ako je x^* rešenje problema 3.9, onda je skup $C = t(x^*)$ maksimalna klika od G sa $|C| = -f(x^*)$.

Drugi način odredjivanja ograničenja za 3.9 je da se koristi činjenica da je kvadratni izraz $x_i x_j = 0$, $\forall (i,j) \in \bar{E}$, s obzirom da za $x_i, x_j \in \{0,1\}$ važi $x_i + x_j \leq 1$ akko je $x_i x_j = 0$. Ograničenja u 3.9 se mogu otkloniti dodavanjem dvostrukog kvadratnog izraza u funkciji cilja, odnosno

$$f(x) = -\sum_{i=1}^{n} x_i + 2 \sum_{(i,j)\in\bar{E}, i>j} x_i x_j = x^T (A_{\bar{G}} - I)x$$

Kvadratni izraz predstavlja (kazneni) penal za prekršaj $x_i x_j = 0$.

Elementi izvan dijagonale matrice A su isti kao elementi matrice susedstva $\bar{G}$. Formulacija 3.9 je unapredjenje za guste grafove. Iz jednakosti problema maksimalne klike i problema stabilnog skupa komplementarnog grafa dobija se:

Teorema 3.4.2 *Problem maksimalnog stabilnog skupa je ekvivalentan sa sledećim problemom neprekidne kvadratne $0-1$ optimizacije*

$$\min f(x) = x^T A x, \quad p.o. \quad x \in \{0,1\}^n \tag{3.10}$$

gde $A = A_G - I$. Ako x^* rešava 3.10 onda skup S koji je definisan sa $S = t(x^*)$ jeste maksimalan stabilan skup od G sa $|S| = -f(x^*)$

Težinski slučaj

Jedna od najjednostavnijih formulacija problema maksimalne težinske klike jeste formulacija preko grana grafa, tzv. edge formulacija.

Definicija 3.4.2 *Neka je dat skup V čvorova i skup E ivica u grafu G, onda je*

$$\max \sum_{i=1}^{n} w_i x_i, \quad p.o. \quad x_i + x_j \leq 1, \forall (i,j) \in \bar{E}, \quad x_i \in \{0,1\}, \quad i = 1, .., n \tag{3.11}$$

Sledi primer formulacije preko analognog problema maksimalnog stabilnog skupa [28]:

Definicija 3.4.3 *Neka je dat skup $\mathcal{S}$ svih maksimalnih stabilnih skupova u grafu G, onda je*

$$\max f(x) = \sum_{i=1}^{n} w_i x_i, \quad p.o. \sum_{i\in S} x_i \leq 1, \forall S \in \mathcal{S}, x \in \{0,1\}, i = 1, .., n \tag{3.12}$$

Vektor x koji predstavlja rešenje Formulacije 3.12, definiše takodje maksimalnu kliku komplementarnog grafa grafu G. Prednost Formulacije 3.12 u odnosu na 3.11 je manja razlika izmedju optimalnih vrednosti u 3.12 i njegove linearne relaksacije. Kako je broj ograničenja u 3.12 eksponencijalan, rešavanje linearne relaksacije 3.12 nije lak problem. Grotchel i drugi [106] su pokazali da je linearna relaksacija problema 3.12 NP-težak zadatak na generalnom skupu grafova. Pokazali su da je sličan problem polinomijalno rešiv na savršenim grafovima [104].

Kasnije su pokazali da je graf savršen akko optimalno rešenje relaksacije od 3.12 uzima celobrojne vrednosti [106].

Data je još jedna formulacija maksimalne kllike u okviru neprekidnog 0-1 programiranja [28]:

$$\min f(x) = x^T A x, \quad p.o. \quad x \in \{0,1\}^n \tag{3.13}$$

gde je $a_{ii} = -w_i$, $i = 1, ..., n$ $a_{ij} = \frac{1}{2}(w_i + w_j)$, $\forall (i,j) \notin E$ i $a_{ij} = 0$, $\forall (i,j) \in E$. Ako je x^* rešenje problema $f(x)$ onda skup definisan sa $C = t(x^*)$ jeste maksimalna težinska klika u G gde je težina $W(S) = -f(x^*)$. Važi relacija izmedju lokalnog minimuma kvadratnog programa i maksimalne klike u grafu:

Teorema 3.4.3 *x je diskretno lokalno rešenje problema 3.13 akko opisuje maksimalnu kliku u G.*

3.5 Formulacije neprekidnog tipa

Maksimalna klika iako predstavlja jedan od najpoznatijih problema kombinatorne optimizacije, to je takodje problem koji je moguće prikazati preko kontinualne funkcije. Funkcija maksimalne klike, osim što je neprekidna, jeste i nekonveksna, odnosno ona pripada klasi nekonveksnih funkcija. Uvodi se klasična definicija nekonveksnog optimizacionog problema koji je dat funkcijom cilja sa ograničenjima:

Definicija 3.5.1 *Kaže se da optimizacioni problem nije konveksan ukoliko važi da je ili f nekonveksna ili su $g_i, i \in \mathcal{I}$ nekonveksni ili su $g_i, i \in \mathcal{E}$ neafini (nelinearni) ili je X nekonveksan i otvoren.*

Jedna od najstarijih kontinualnih formulacija problema MC jeste formulacija Motzkina i Strausa [178]. Ona pokazuje vezu izmedju globalnog maksimuma Lagranžijana u grafu na standardnom simpleksu i kardinalnosti maksimalne klike u G. Formulacija takodje pokazuje vezu izmedju problema standardnog kvadratnog programiranja i maksimalne klike dokazujući slabiju verziju fundamentalne teoreme Turana [214]. Data je preko neprekidne nelinearne funkcije i na nju se oslanjaju novi pravci d.c. programiranja [12]. U literaturi se mogu pronaći mnoge verzije ove teoreme s obzirom da je ova teorema inspirisala mnoge autore širom sveta jer predstavlja vezu dva naizgled različita pristupa u optimizaciji: diskretnog i kontinualnog. Prvo se može videti savremenija verzija koju je dao Butenko [46], a nakon toga se može videti originalna verzija teoreme. Dokazi se mogu pogledati u datoj literaturi. Teorema Motzkin-Straus u jednoj od novijih verzija je početna tačka istraživanja ove disertacije, stoga se može reći da je teorema jedan od fundamentalnih pojmova od koga se krenulo u istraživanje.
Neka je dat graf $G = (V, E)$ kao neorijentisan i neteżinski graf i neka je Δ standardni simpleks u n-dimenzionalnom Euklidskom prostoru R^n;
neka je Δ_D podskup od Δ, u odnosu na pridruženi skup $D \subseteq V$.

$$\Delta = \{x \in R^n : x_i \geq 0, \forall i \in V, e^T x = 1\}$$

gde je vektor e definisan kao vektor adekvatne dužine a sastoji se od jedinica. Za njega važi da je $e^T x = \sum_{i=1}^{n} x_i$.
Posmatra se kvadratna funkcija koja se zove Lagranžijan

$$g(x) = x^T A_G x \tag{3.14}$$

gde je $A_G = (a_{ij})_{i,j \in V}$ matrica susedstva od G, simetrična matrica $n \times n$, $a_{ij} = 1, (i,j) \in E \quad a_{ij} = 0, (i,j) \notin E$. Neka je x^* globalni maksimizator g u Δ. Red klike u G je dat formulom

$\omega(G) = \frac{1}{1-g(x^*)} \geq \frac{1}{1-g(x)}, \forall x \in \Delta$. Pokazano je da je podskup čvorova C maksimum klike u G akko je karakterističan vektor x^C, globalni maksimizator od g na Δ, a on je definisan sa ${x_i}^C = \frac{1}{|C|}$ ako $i \in C$ i ${x_i}^C = 0$ inače. Dakle, vektor je globalni maksimizator od funkcije g na simpleksu.

Dokaz se može videti u [46].

Originalna Motzkin-Straus teorema glasi:

Teorema 3.5.1 *Neka je k red maksimalnog kompletnog podgrafa koji se sadrži u G. Tada važi*

$$f(G) = \frac{1}{2}(1 - \frac{1}{k})$$

Dokaz se može videti u [178].

Uslovi optimalnosti Motzkin-Straus programa su studirani i istraživani što je dovelo do uvodjenja nove parametrizacije u kvadratnom programu klike. Sos i Straus [205] su dalje generalizovali istu teoremu u hipergrafovima.

U kasnijim radovima, teorema Motzkin-Straus se menjala u smislu lokalnih maksimizatora funkcije g na raznim politopima. Jedan od problema koji se javio u originalnoj formulaciji je problem prividnih rešenja, odnosno maksimizatora koji nisu dati u formi karakterističnih vektora. Takva rešenja ometaju izdvajanje čvorova iz klike. Na osnovu originalne teoreme istraživači su razvili metode koji se oslanjaju na nju ali su pokazali nove karakteristike i mogućnosti rešavanja maksimalne klike u okviru neprekidnog domena. Pardalos i Xue [189] su razvili proces iterativne pretrage klike. Bomze i drugi [27] su koristili algoritme relaksacije da odrede veličinu klike. Gibbons, Hearn i Pardalos [92] su dali parametarsku formulaciju teoreme Motzkin-Straus. Gibons [93] je dao karakteristike uslova optimalnosti prvog i drugog reda Motzkin-Straus teoreme, i proširio je rezultat koji je omogućio karakterizaciju maksimalne klike u smislu lokalnih rešenja.

Problem prividnih rešenja su rešavali Bomze i drugi [28]. Regularizovali su funkciju g: $\widehat{g} = x^T A_G x + \frac{1}{2}x^T x$, $\widehat{A_G} = A_G + \frac{1}{2}I$, gde je I matrica identiteta.

Teorema 3.5.2 *Neka je C podskup čvorova grafa G i neka je x^C njegov karakterističan vektor. Tada važi sledeće tvrdjenje:*

C je maksimum klike (engl. Maximum clique) od G akko x^C jeste globalni maksimizator funkcije $\widehat{g}$ na simpleksu Δ. U ovom slučaju je $\omega(G) = \frac{1}{2(1-\widehat{g}(x^C))}$.
C je maksimalna klika u G akko x^C jeste lokalni maksimizator funkcije $\widehat{g}$ na simpleksu Δ.
Svi lokalni i globalni maksimizatori x od $\widehat{g}$ na simpleksu Δ su striktni maksimizatori, dati formom $x = x^C$ za neko $C \subseteq V$.

Kao što se vidi, dokazano je da su svi maksimizatori od $\widehat{g}$ na simpleksu striktni i da su oni karakteristični vektori maksimalne klike u grafu i da postoji preslikavanje sa klike na globalne i lokalne maksimizatore. Ovim se rešio problem prividnih rešenja. Na osnovu ove formulacije, Gibons [28] je generalizovao problem na težinski slučaj.

Reformulisana je Motzkin-Straus teorema na problem minimizacije tako što se posmatra funkcija $f(x) = x^T(I + A_{\bar{G}})x$, gde je $A_{\bar{G}}$ matrica susedstva komplementa $\bar{G}$. Ako je x^* globalni minimizator f u Δ, onda važi $\omega(G) = \frac{1}{f(x^*)}$.

3.6 Pregled radova iz oblasti MC

Najpoznatiji skup koji je organizovan za rešavanje klike je održan 1993. godine pod nazivom Klike, Bojenje, Zadovoljivost (engl. Cliques, coloring, satisfiability) na kojem je cilj bio izrada što bolje metode za odredjivanje maksimalne klike. U toku nadmetanja su pojedini učesnici predlagali kolekcije grafova koji su poslužili za proveru algoritama nastalih na tom nadmetanju. Mnogi od tih grafova se i danas koriste za proveru karakteristika novih metoda i poredjenje sa drugim metodama. Na tom nadmetanju je predloženo petnaestak heuristika za odredivanje

približnog rešenja. Osim toga je od sedamdesetih godina pa do današnjih dana predložen veliki broj metoda za odredjivanje tačnog rešenja problema. Isto tako je većina metaheurističkih metoda iskorišćena za odredjivanje rešenja: GRASP, Simulirano kaljenje, Tabu pretraživanje, Genetski algoritmi itd.

3.6.1 Egzaktni algoritmi

Svi algoritmi koji se pominju su kreirani za pronalaženje maksimalne klike u grafu. Ipak, da bi se rešio problem maksimalne klike potrebno je da se pronadje broj klike i maksimalna klika koja je sa njim u vezi. U literaturi postoje mnogi egzaktni algoritmi za pronalaženje maksimalne klike i srodnih problema. Mnogi od njih su varijacije metode grananja i ogradjivanja (engl. Branch and bound) i oni se mogu definisati preko različitih tehnika za odredjivanje donje i gornje granice ili preko odgovarajućih strategija grananja. Harary i Ross [117] su 1957. objavili prvi algoritam za prebrojavanje svih klika u grafu. Njihov rad je bio motivisan primenama u sociometriji. Ideja te metode je da se redukuje problem sa generalnih grafova na specijalan slučaj grafova koji sadrže najviše tri klike i onda da se reši problem za taj specijalan slučaj. Njihov rad su pratili mnogi drugi algoritmi. Paull, Unger i Marcus [164, 196] su predložili algoritme za minimizovanje broja stavova u sekvencijalnom zamenjivanju funkcija. Rad autora Bonner-a [29] je motivisan problemom klasterovanja. Bednarek i Taulbe [20] su tragali za maksimalnim lancima u skupu sa datim binarnim relacijama. Iako su svi ovi pristupi dizajnirani da reše probleme za različite aplikacije, njihova prvobitna ideja je da prebroje sve klike u nekom grafu. Razvoj računarske tehnologije 60-tih godina prošlog veka je omogućio da se testiraju algoritmi na grafovima većih dimenzija. Kao rezultat, ranih 70-tih godina mnogi novi algoritmi prebrojavanja su predloženi i testirani. Najvažniji medju njima je bio algoritam pod nazivom Pretraživanje unatrag (engl. Backtracking method) autora Bron-a i Kerbosch-a [35]. Prednost njihovog pristupa je uključivala zahtev za polinomijalnom memorijom i isključivanje mogućnosti za generisanjem iste klike dva puta. Algoritam je uspešno testiran na grafovima sa 10 do 50 čvorova i sa gustinom ivica u rangu izmedju 10% i 95%. Modifikaciju ovog pristupa uradio je Tomita [212] i objasnio da vremenska kompleksnost od $O(3^{n/3})$ jeste najbolje moguće vreme za koje algoritam prebrojavanja može da se izvede s obzirom na postojanje grafova sa $3^{n/3}$ maksimalnih klika [177]. Algoritam Bron-Kerbosch je pokazao da je efikasan sa grafovima koji se javljaju u trodimenzionoj molekularnoj strukturi [89].

Autori Tarjan i Trojanowski [211] su predložili Rekurzivan algoritam za problem maksimalnog stabilnog skupa sa vremenskom kompleksnosti od $O(2^{n/3})$. Kasnije, ovaj je rezultat poboljšao Robson [200], koji je modifikovao algoritam Tarjan-a i Trojanowskog kako bi dobio vremensku kompleksnost od $O(2^{0.276n})$. Drugi važan algoritam su predložili Balas i Yu 1986. godine [15]. Koristili su implicitnu enumeraciju i bili su u mogućnosti da izračunaju maksimalne klike u grafovima do 400 čvorova i 30,000 ivica.

Carraghan i Pardalos [51] su predložili drugi implicitni algoritam nabrajanja za problem maksimalne klike koji su bazirali na istraživanju čvorova u poretku koji je u vezi sa ne opadajućim poretkom stepena čvorova. Pristup se dokazao kao vrlo efikasan, posebno za retke grafove. Javno dostupna implementacija ovog algoritma služi kao dobra praksa prilikom poredjenja sa drugim algoritmima [145]. Njihov algoritam izbegava nabrajaje svih klika i umesto toga radi sa znatno redukovanom parcijalnom enumeracijom. Zanemarivanje u enumeraciji se postiže strategijom Redukcije koja onemogućava pretragu prostora značajno. Algoritam se izvršava tako što se u svakom koraku i primenjuje algoritam obilaska u dubinu od čvora v_i pri čemu je cilj pronaći najveću kliku koja sadrži taj čvor. U svakoj iteraciji obilaska u dubinu, algoritam uporedjuje broj preostalih čvorova koji potencijalno mogu da učestvuju u kliki koja sadrži čvor v_i, u odnosu na veličinu najveće klike koja je do sada pronadjena. Ukoliko je broj na koji se naidje manji, algoritam se prekida, tj. odustaje se od dalje pretrage, odnosno pretraga se redukuje (engl. Pruning technique).

Skoro je Ostergard [182] predložio algoritam grananja i ogradjivanja koji analizira čvorove u poretku definisanom putem bojenja, te je time uveo novu strategiju odbacivanja pojedinih čvorova. Promenio je poredak pretrage u odnosu na prethodni algoritam. Uporedio je izvodjenje ovog algoritma sa nekoliko drugih pristupa na slučajnim grafovima i na DIMACS-ovim primerima, i dokazao je da je njegov algoritam superiorniji. I on je koristio novu poboljšanu tehniku Redukcije uz mehanizam pomoćnog zapisivanja i pamćenja pretrage. Strategija Redukcije koju

su autori prikazali u svom algoritmu je blisko povezana sa redosledom obradjivanja čvorova, tako da je sekvencijalnost jedno od svojstva ovog algoritma.
algoritmi koji su predloženi u [195] takodje koriste dodatnu tehniku Redukcije, jednostavni su i izbegavaju sekvencijalnost u izvršavanju. Prva metoda koju su autori predložili je egzaktna metoda pronalaženja najveće od svih maksimalnih klika koje sadrže neki čvor. Tokom pretrage najveće klike koja sadrži dati čvor, čvorovi koji ne mogu da formiraju klike veće od trenutne maksimalne klike se redukuju po hijerarhijskoj osnovi. Autori su uveli promenljivu max koja čuva veličinu pronadjene maksimalne klike do sada. Inicijalno, promenljiva se postavlja na donju granicu lb koja je ulazni parametar. Kada se pronadje klika maksimalne veličine, algoritam se zaustavlja. Kako bi se pronašla maksimalna klika koja sadrži čvor v_i, dovoljno je da se razmatraju susedi od v_i. U okviru glavne procedure, za svaki čvor $v_i \in V$ se generiše skup $U \subseteq N(v_i)$, skup čvorova u okolini čvora v_i koji preživljavaju Redukciju. Zatim se prolazi kroz svaku relevantnu kliku koja sadrži v_i na rekurzivan način i dolazi se do najveće. Autori su koristili promenljivu *velicina* da odrede veličinu klike koju pronalaze u svakoj iteraciji rekurzije. Kako su startovali sa klikom koja je veličine jednog čvora, promenljivu *velicina* su postavili na jedan inicijalno. Ovaj algoritam se sastoji od nekoliko koraka strategije Redukcije. Redukcija 1 bira čvorove koji striktno imaju manje suseda nego što je veličina trenutne maksimalne klike. Ovi čvorovi mogu biti ignorisani jer njena veličina ne bi bila veća od max veličine. Redukcija 2 se koristi radi izbegavanja ponovnog izračunavanja prethodno pronadjenih klika. Uključuju se samo oni čvorovi iz liste suseda U za koje najveća klika (koja ih sadrži) nije pronadjena. Redukcija 3 isključuje čvorove $v_j \in N(i)$ koji imaju stepen manji od trenutne vrednosti max. Redukcija 4 proverava slučaj da kada su svi čvorovi u U dodati u kliku, njena veličina neće premašiti veličinu do sada najveće pronadjene klike, max. Redukcija 5 smanjuje broj uporedjivanja potrebnih da se generišu intersekcije.

Kao i mnogi problemi kombinatorne optimizacije, i problemi maksimalne klike i maksimalnog stabilnog skupa se mogu formulisati kao celobrojni programi. Neki od moćnih alata koji služe rešavanju celobrojnih programa su bazirani na dobrim optimizacijskim paketima, kao npr. ILOG CPLEX [129] i Dash Optimization Xpress [64], koji kombinuju algoritam grananja i ogradjivanja sa Metodom odsecanja ravni, efikasnim pred-procesiranjem, brzim heuristikama i sofisticiranim tehnikama dekompozicije u cilju nalaženja tačnog rešenja.

3.6.2 Heuristike i metaheuristike

Iako su egzaktni pristupi omogućavali optimalno rešenje, postali su nepraktični i prespori čak i za grafove sa nekoliko stotina čvorova. Kada se obradjuje problem maksimalne klike na velikim grafovima, egzaktni algoritmi se ne mogu primeniti, a heuristike postaju jedina dostupna opcija. Jedna od osnovnih (meta)heuristika za problem maksimalne klike jeste Sekvencijalna pohlepna heuristika (engl. Sequential greedy heuristics, SGH) [143] koja ponavlja dodavanje čvorova u skup u kome su svi postojeći čvorovi povezani medjusobno i koja pomera čvor iz skupa klike ukoliko on nije povezan sa čvorovima u skupu.
Pored SGH kao jednostavan heuristički algoritam izdvaja se i Multistart lokalna pretraga (engl. Multistart local search, MLS) [163].
Što se tiče VNS [168] algoritma za problem MC, on kombinuje pohlepnu pretragu sa testiranjem povezanih čvorova tokom spusta. Koriste se tri tipa okolina: "drop", "add" i "interchange".

Primer Simuliranog kaljenja za problem maksimalne klike koristi pristup funkcije penala, što je opisano u tekstu autora Aarts-a i Korst-a [2]. Oni koriste skup svih mogućih podskupova čvorova kao prostor rešenja, dok je funkcija cilja u formi $f(V') = |V'| - \lambda|E'|$, gde je $V' \subseteq V$ podskup čvorova, a $E' \subset V' \times V'$, skup ivica u $G(V')$. Homer i Peinado [125] su implementirali varijaciju algoritma Aarts-a i Korst-a sa jednostavnim rasporedom hladjenja i uporedili su izvršavanje sa izvršavanjem drugih heuristika za problem maksimalne klike. Nakon eksperimentisanja na grafovima sa 70,000 čvorova, zaključili su da je pristup simuliranog kaljenja superiorniji u odnosu na ostale kompetitivne heuristike nad razmatranim instancama.

Mnoge verzije Neuralnih mreža se mogu primeniti na problem maksimalne klike u grafu što su pokazali neki istraživači još od kraja 80-tih godina prošlog veka. Efikasnost ranih poduhvata je teško evaluirati s obzirom na nedostatak eksperimentalnih rezultata. Grossman [102] je razmatrao diskretnu verziju Hopfield-ovog modela za maksimalnu kliku. Rezultati računskih eksperimenata sa ovim pristupom na DIMACS testnim primerima su bili zadovoljavajući, mada

u tom momentu, bili su inferiorniji u odnosu na druge računarske metode, kao npr. simulirano kaljenje. Jagota [131] je predložio nekoliko različitih diskretnih i kontinualnih verzija modela Hopfield-a za problem maksimalne klike. Kasnije, u jednom radu [132] autori su poboljšali algoritam kako bi dobili značajno veću kliku nego što je ona koja je pronadjena jednostavnijom heuristikom, a koja samo malo brže radi.
Raniji pokušaji da se primeni genetski algoritam na maksimalnu kliku i maksimalni stabilni skup datiraju od 90-tih godina prošlog veka. Mnoge uspešne implementacije se pojavljuju u literaturi [124]. Takodje, mnogi genetski algoritmi se lako paralelizuju.
GRASP se primenjivao uspešno na razne probleme kombinatorne optimizacije uključujući problem maksimalne klike [84]. U okviru algoritma predlaže se dodatak koji se naziva povezivanje puta (engl. Path relinking) [95] kako bi se obezbedilo izvodjenje heuristike povezivanjem rešenja dobrog kvaliteta sa putanjom srednjih dopustivih tačaka. U mnogim slučajevima neke od ovih dopustivih tačaka omogućavaju bolji kvalitet od rešenja koja se koriste kao krajnje tačke u putanji. GRASP procedura zajedno sa procedurom putanje se može primeniti za maksimalnu kliku, maksimalan stabilan skup i problem bojenja grafova.
Razne verzije Tabu pretrage su takodje uspešno primenjene na probleme maksimalne klike i maksimalnog stabilnog skupa [18].
Heuristika koja je predložena u [195] jeste heuristika koja ispituje da li sve relevantne klike sadrže svaki čvor. Heuristika uzima u obzir samo najveće stepene u svakoj iteraciji, umesto rekurzivnog biranja, što se pokazalo da se pretraga dosta brže izvršava.

Pregled još nekoliko značajnih algoritama:
Fazna lokalna pretraga (engl. Phased local search) [197] je stohastički algoritam lokalne pretrage koji propisuje podalgoritme. Oni se izvršavaju izmedju sekvenci iterativnog poboljšanja, tokom kojih se pogodni čvorovi dodaju trenutnoj kliki i plato pretrage gde se čvorovi trenutne klike zamenjuju sa čvorovima koja se ne sadrže u trenutnoj kliki.
Reaktivna lokalna pretraga (engl. Reactive local search) [18] je algoritam koji je proizašao iz reaktivne tabu pretrage; on je unapredjena tabu metoda koja automatski adaptira mandat tabu parametra.
Duboka adaptivna pohlepna pretraga (engl. Deep adaptive greedy search) [103] koristi iterativnu pohlepnu konstrukciju sa težinama čvorova.
K-opt algoritam [147] je baziran na konceptualno jednostavnoj proceduri koja koristi korake elementarne pretrage i u kojoj se čvor dodaje ili uklanja iz trenutne klike.
Kooperativna metoda lokalne pretrage (engl. CLS) [198] je paralelizovana hiper-heuristika koja je razvijena za MC problem. CLS koristi dva dela u okviru metode: dela za kooperativnu heuristiku niskog nivoa koja se kreće izmedju sekvenci iterativnih poboljšanja tokom koje se pogodni čvorovi dodaju trenutnoj kliki i dela plato pretrage gde se čvorovi trenutne klike zamenjuju sa čvorovima koji nisu u kliki.

Heuristike bazirane na neprekidnoj optimizaciji

Kao što je rečeno, neprekidni pristup problemima kombinatorne optimizacije postaje posebno zanimljiv. Razvijanje neprekidno baziranih heuristika za probleme maksimalne klike i maksimalnog stabilnog skupa se oslanja na teoremu i formulaciju Motzkin-Straus [178] koja povezuje broj klike u grafu sa kvadratnim programom. Ova formulacija se dokazuje i diskutuje detaljno u odeljku Formulacije neprekidnog tipa. Pored toga, dokazuju se još neke neprekidne formulacije razmatranih problema, a suština rada je da se razvijaju i testiraju algoritmi koji su bazirani na skupu neprekidnih formulacija. U skorašnje vreme otkrivene su i razvijane semidefinitne tehnike programiranja, koje su u suštini neprekidno formulisane. Izvanredan rezultat za MC su dali Goemans i Williamson [97] i on služi kao glavni korak napred u razvoju aproksimativnih algoritama koji dokazuje specijalnu važnost semidefinitnog programiranja. Burer i drugi [44] su uzeli u obzir i relaksacije ranga 1 i ranga 2 Lovasz-ovog semidefinitnog programa [159] i time su dobili dve neprekidne optimizacijske formulacije za probleme maksimalne klike i maksimalnog stabilnog skupa. Na osnovu ovih formulacija, razvili su i testirali nove heuristike za pronalaženje velikih stabilnih skupova.
Klasični algoritmi konveksne optimizacije se odbacuju zbog slabe efikasnosti, s obzirom da postoji mogućnost da se rešenje zaglavi u lokalnom ekstremumu ili u kritičnoj tački. Zbog toga se opisuju dva algoritma koji su pogodni kod rešavanja nekonveksnih problema predstavljenih

preko više formulacija. Strekalovski [206] je razvio pristup za rešavanje neprekidnih nekonveksnih problema i primenio ga na problem maksimalne klike, s obzirom da je problem maksimalne klike nekonveksan problem. Mnogi optimizacioni problemi koji spadaju pod klasu nekonveksnih se mogu rešiti preko d.c. (difference-convex) funkcija, odnosno preko razlika dve konveksne funkcije. D.C. klasa funkcija poseduje nekoliko važnih osobina kao:

- $DC(R^n)$ je vektorski prostor, generisan preko poznate klase - konusa konveksnih funkcija.
- $DC(R^n)$ sadrži klase prostora $C^2(R^n)$ kao i stepene, trigonometrijske funkcije, itd.
- Bilo koja neprekidna funkcija duž kompakta na (R^n) se može aproksimirati preko nekih d.c. funkcija sa proizvoljnom tačnošću.

Strekalovski predlaže nov pristup koji se sastoji od globalnih uslova za optimalnost ili GOC (Global Optimality Conditions) za klasu d.c. problema. Polazi se od neprekidne formulacije MC problema koju su dali Motzkin i Straus tako što se on formuliše kao d.c. nedefinitni problem kvadratne maksimizacije preko kanoničnog simpleksa na kom se primenjuju uslovi globalne optimizacije. Predložena je analitična provera algoritma globalne pretrage, zatim sledi algoritam globalne pretrage (engl. Global Search algorithm, GSA), a nakon toga sledi linearizovan algoritam koji prati problem.

Definicija 3.6.1 *Neka su* $X = (x_1, x_2, .., x_n)$ *i* $Y = (y_1, y_2, ..., y_n)$ *vektori, tada je skalarni proizvod vektora (engl. Dot product) dat sa* $\langle X, Y\rangle = X \cdot Y = \sum x_i y_i$

Strekalovski je formulisao problem tako što je pošao od generalnog d.c. problema maksimizacije: $F(x) = \frac{1}{2}\langle x, Ax\rangle \uparrow \max$, $x \in S$, gde je S kanonski simpleks, a matrica A je jednaka $A = A_G + \frac{1}{2}I_n$, gde su matrica A_G matrica susedstva, a matrica $I_n$matrica identiteta. $F(x) = f(x) - g(x) \uparrow \max$, $x \in D$, gde su $f()$ i $g()$ konveksne funkcije na konveksnom skupu $D \subset R^n$

Poznato je da matrica A koja je nedefinitna može biti predstavljena kao razlika dve pozitivno definitne matrice: $A = A_1 - A_2$.

Zadatak se svodi na rešavanje nekonveksnog problema definisanog kao razlika dve konveksne funkcije: $F(x) = f(x) - g(x) \uparrow \max, x \in D$ [207]. Dakle, zadatak koji pronalazi maksimalnu kliku je da se reši nekonveksna funkcija koja je aproksimacija razlike dve konveksne funkcije [151]. Ukoliko se pronalazi ekstremum konveksne funkcije, može da se pronadje i približni ekstremum nekonveksnog zadatka.

Butenko [48] je pošao od sličnog principa i prikazao je formulacije za rešavanje problema stabilnog skupa i maksimalne klike: 0-1 kvadratna formulacija bez ograničenja, kao i nekonveksna formulacija.

Pretraga se vrši sekvencijalno po formulacijama, ali i simultano po formulacijama. Formulacije MC su date preko komplementarnog problema stabilnog skupa kao neprekidne formulacije:

$\alpha(G) = \max(e^T x - \frac{1}{2}x^T A_G x)$, $x \in [0,1]^n$ i $1 - \frac{1}{\alpha(G)} = \max x^T A_{\overline{G}} x$, $x \in S$ gde je $S = \{X \in R^{|V|} : e^T x = 1, x \geq 0\}$ i e je vektor sa svim komponentama jednakim jedinici. U okviru ovih formulacija, neprekidan dopustivi region $x \in [0,1]^n$ može biti zamenjen sa diskretnim $x \in \{0,1\}^n$ bez promene optimalne vrednosti funkcije cilja. Dopustivi region se može diskretizovati preko

$S' = \{y = \frac{x}{e^T x} : x \in \{0,1\}^n, x \neq 0\}$. Nakon transformacija dobijaju se diskretne formulacije

$\alpha(G) = e^T x - \frac{1}{2}x^T A_G x$, gde $x \in \{0,1\}^n$ i $1 - \frac{1}{\alpha(G)} = x^T A_{\overline{G}} x$, gde $x \in S'$.

Butenko je koristio neophodne funkcije prelaza iz formulacije u formulaciju $h_{(12)}(x) = 0$, ukoliko je $x = 0$, $h_{(12)}(x) = \frac{x}{(e^T)x}$, ukoliko je $x \neq 0$

Okolina nekog čvora se posmatra kao skup čvorova na Hamingovom rastojanju 1 od čvora x. Dakle, autor Butenko je krenuo od neprekidnih formulacija da bi ih nakon diskretizovanja sveo na kombinatorne $0-1$ kvadratne formulacije.

4

Metoda promenljivih okolina i formulacija

Metoda promenljivih okolina je metaheuristika, koja je predstavljena devedesetih godina prošlog veka [168] nakon čega je doživela mnogo primena i samim tim ekstenzija. VNS metaheuristika zasnovana je na tri osnovne činjenice:

1. Lokalni minimum u odnosu na jednu okolinu ne mora biti i lokalni minimum u odnosu na neku drugu okolinu
2. Globalni minimum je lokalni minimum u odnosu na sve okoline
3. Za većinu problema lokalni minimumi u odnosu na razne okoline su medjusobno bliski

Za razliku od drugih metaheuristika, osnovna verzija VNS metode je jednostavna i zahteva malo parametara, tačnije, osnovni i jedini parametar je k_{max}, gde je k_{max} redni broj najšire okoline u kojoj se obavlja razmrdavanje. VNS omogućava da se dostignu korektna rešenja, jednostavnijim načinom u odnosu na ostale metode, dok je implementacija vrlo jasna i efikasna. Ideje koje su prethodile VNS metodi jesu Metoda promenljive metrike koji se koristi za rešavanje kontinualnih problema i lokalna pretraga za rešavanje problema kombinatorne i kontinualne optimizacije. **Metodu promenljive metrike** (engl. Variable metric method) su predložili Davidon [110] i Fletcher i Powell [110] 1959. i 1963. respektivno. U svakoj iteraciji koja je predstavljena pretragom pravca (pravac najstrmijeg spusta) dolazi do promene metrike i okolina. Pretraga pravaca se prilagodjava oblicima funkcija. U prvoj iteraciji koristi se jedinična Euklidska lopta u prostoru n dimenzija i odredjuje se **pravac najstrmijeg spusta** (engl. Steepest descent). Sledeće iteracije su odredjene elipsoidom, dok je pravac najstrmijeg spusta odredjen u odnosu na novu metriku koja rezultira iz linearnih transformacija. Svrha je da se izgrade iterativno dobre aproksimacije u odnosu na inverz Hesian matrice A^{-1} od f, odnosno da se konstruiše sekvenca matrica H_i sa karakteristikom

$$\lim_{i \to \infty} H_i = A^{-1}$$

U okviru konveksnog kvadratnog programiranja, kraj se dostiže nakon n iteracija, a ne nakon beskonačno iteracija, čime se postiže Newtonova pretraga pravca. Nije neophodno u svakoj iteraciji pronaći inverz Hesiana, samim tim nije neophodno računati matricu parcijalnih izvoda drugog reda. U svakoj iteraciji vrši se korekcija matrice H_i preko korektivnog faktora U, $H_i+1 = H_i + U$.

Pored Metode promenljive metrike, na kreiranje VNS metode je uticala takodje i lokalna pretraga. Ona se sastoji od izbora inicijalnog rešenja x, pronalaženja smera spusta od x, u okviru okoline $N(x)$ i pomeranja ka minimumu $f(x)$ u istom smeru. Ukoliko ne postoji smer spusta, heuristika prekida rad; inače se iteracije ponavljaju sve dok postoji. Smer najstrmijeg spusta se takodje označava kao **najbolje poboljšanje** (engl. Best improvement). $N(x)$ se definiše za sve $x \in X$, a kod problema kombinatorne optimizacije obično se sastoji od svih vektora koji se formiraju od x uz neke modifikacije. Iako je vremenski dosta iscrpljujuće, uglavnom se istražuje

Algoritam 1 Promenljiva metrika(x)

Neka je $x \in R^n$ inicijalno rešenje;
$H \leftarrow I; g \leftarrow -\nabla f(x)$;
for $i = 1$ to n **do**
  $\alpha^* \leftarrow \arg\min_\alpha f(x + \alpha Hg)$;
  $x \leftarrow x + \alpha^* Hg$;
  $g \leftarrow -\nabla f(x)$;
  $H \leftarrow H + U$;
end for

kompletna okolina $N(x)$ (vidi Algoritam 3). **Prvo poboljšanje** (engl. First improvement) se uvodi kao alternativa vremenski zahtevnom pretraživanju kompletne okoline (vidi Algoritam 2). Tada su vektori $x_i \in N(x)$ nabrojani sistematično, a pomeraj se vrši čim se smer spusta pronadje.

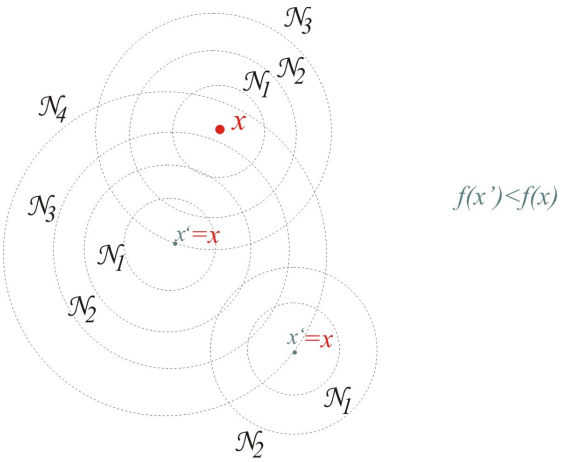
Algoritam 2 Prvo poboljšanje

repeat
  $x' \leftarrow x; i \leftarrow 0$;
  repeat
    $i \leftarrow i + 1$;
    $x \leftarrow argmin\{f(x), f(x_i)\}, x_i \in N(x)$
  until $(f(x) < f(x_i))$ ili je $i = |N(x)|$;
until $(f(x) \geq f(x'))$;

Algoritam 3 Najbolje poboljšanje

repeat
  $x' \leftarrow x$;
  $x \leftarrow argmin_{y \in N(x)} f(y)$
until $(f(x) \geq f(x'))$;

U ovom poglavlju predstavljene su neke od varijanti Metode promenljivih okolina, sa zadržavanjem na Metodi promena formulacija.



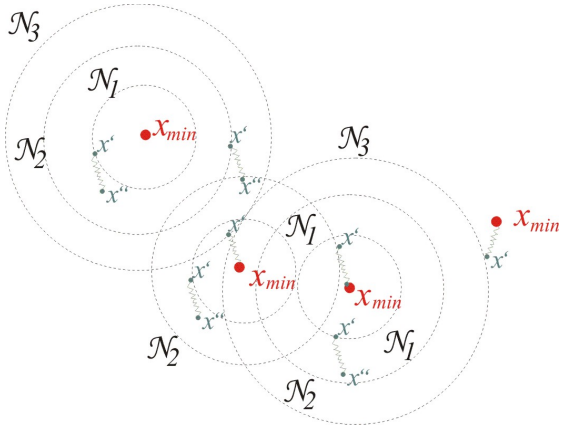
Slika 4.1: Promena okolina [67]

4.1 Metoda spusta kroz promenljive okoline

Metoda spusta kroz promenljive okoline [110] (engl. Variable neighborhood descent, VND) bazira se na tome da neko rešenje x ne mora biti istovremeno lokalni optimum u dve razne okoline. Drugim rečima, ako su $N_1(x)$ i $N_2(x)$ dve okoline rešenja x, može se desiti da za svako rešenje $x' \in N_1(x)$ važi da je $f(x') \geq f(x)$, a da istovremeno postoji rešenje $x'' \in N_2(x)$ takvo da je $f(x'') < f(x)$. Da bi se realizovao spust kroz promenljive okoline, u prostoru rešenja mora se definisati konačni skup okolina $N_k, k = 1, 2, ..., k_{max}$, tako da je $N_k(x) \subset X$ skup rešenja koja obrazuju k-tu okolinu rešenja x (vidi sliku 4.1).

Spust počinje od nekog početnog rešenja x izabranog na slučajan način ili primenom neke konstruktivne heuristike. Zatim se od njega, obavlja lokalno pretraživanje u okolini $N_1(x)$. U toku te lokalne pretrage dobija se neki lokalni optimum x_1 što je lokalni optimum u odnosu na okolinu $N_1(x)$. Zatim se proverava postupak kroz ispitivanje rešenja u okolini $N_k(x_{k-1}), k = 2, 3, ..., k_{max}$, pri čemu je x_{k-1} lokalni optimum dobijen pretragom okoline N_{k-1}. Ukoliko se ne pronadje rešenje ni u jednoj mogućoj okolini rešenje x_1 je istovremeno lokalni optimum u odnosu na sve definisane okoline, pa prema tome i približno rešenje problema.

Ako se u nekoj od okolina $N_k(x_{k-1})$ pronadje rešenje x_k bolje od trenutnog rešenja, onda se vrši pomeranje u x_k i produžava se spust lokalnom pretragom u okolini N_1.



Slika 4.2: Promena okolina i lokalna pretraga [67]

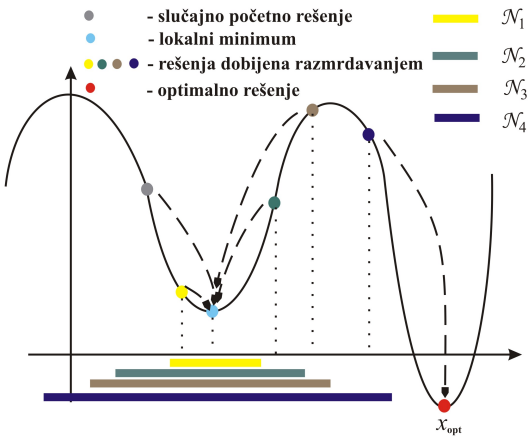
Algoritam 4 Metoda spusta kroz promenljive okoline

Inicijalizacija
Odredi kolekciju okolina $N_k, k = 1, 2, ..., k_{max}$ u prostoru rešenja;
Odredi početno rešenje, $x \in X$;
Postavi k na 1, $(k \leftarrow 1)$;
repeat
1. Istraživanje okoline: Pronadji najbolje rešenje $x' \in N_k(x)$;
2. Da li se pomeriti? Ako je $f(x') < f(x)$ pomeri se u $x'(x \leftarrow x')$ i postavi k na 1 $(k \leftarrow 1)$; U suprotnom uvećaj k za 1 $(k \leftarrow k + 1)$;

until $k > k_{max}$;

Kod ove metode praksa je da se za k_{max} uzima relativno mali ceo broj (2, 3, 4). Kompletno istraživanje okoline može biti vrlo zahtevno (vremenski) ili su rešenja takva po prirodi da je utvrdjivanje kvaliteta svakog pojedinačnog rešenja zahtevan proces. Po prethodno ispisanom pseudokodu, u svakoj okolini se pronalazi najbolje rešenje, tj. primenjuje se strategija najbolje popravke (engl. Best improvement). Istraživanje okoline se može prekinuti u trenutku kada se pronadje prvo bolje rešenje, tj. tako da se primenjuje strategija prve popravke (engl. First improvement).

4.2 Osnovna varijanta metode promenljivih okolina



Slika 4.3: Metoda promenljivih okolina [67]

Metoda promenljivih okolina [110] (engl. Variable neighborhood search, VNS) sistematično ispituje prostor rešenja. Da bi se to dobro izvelo u prostoru rešenja, najčešće se definiše neka metrika (rastojanje) ρ. Zatim se u prostoru rešenja definiše konačan skup okolina $N_k, k = 1, 2, ..., k_{max}$. Kaže se da je $x \in X$ proizvoljno rešenje, i da je $N_k(x) \subset X$ skup rešenja koja se nalaze u k-toj okolini rešenja x. U osnovi, okolinu $N_k(x)$ čine sva rešenja koja se nalaze na rastojanju k ili na većem od k, od rešenja x:

$$N_k(x) = \{x' \in X | \rho(x, x') \leq k\}$$

Metoda se izvršava na taj način što se u trenutku nalaženja nekog lokalnog optimuma, izvršava pomeranje do odredjenog rešenja (koje može biti i vrlo loše) i nakon toga pokušava se od tog rešenja stići do nekog drugog lokalnog optimuma (vidi sliku 4.3). Pomeranjem do rešenja koje se nalazi relativno daleko od trenutnog rešenja, postiže se sistematično pretraživanje prostora rešenja. Zadržavanjem u trenutno najboljem rešenju, u slučaju kada lokalno pretraživanje nakon razmrdavanja nije odvelo do boljeg rešenja, smanjuje se mogućnost nepotrebnog lutanja u prostoru rešenja. Kao lokalno pretraživanje se obično koristi standardno lokalno pretraživanje. Okolina koja se koristi u lokalnom pretraživanju, je zapravo N_1 okolina prirodno indukovana uvedenom metrikom (vidi Algoritam 5).

Algoritam 5 Metoda promenljivih okolina

1. Inicijalizacija. Odredi adekvatan skup okolina $N_k, k = 1, 2, ..., k_{max}$ i odredi početno rešenje $x \in X$;
Izaberi kriterijum za prekid izračunavanja
repeat
a) Postavi k na 1 ($k \leftarrow 1$);

repeat
- Razmrdavanje: Izaberi na slučajan način rešenje x' iz $k$te okoline rešenja x ($x' \in N_k(x)$);
- Lokalno pretraživanje. Primeni neko lokalno pretraživanje polazeći od rešenja x'; Označi sa x'' tako dobijeni lokalni optimum;
- Da li se pomeriti? Ako je $f(x'') < f(x)$ pomeri se u $x''(x \leftarrow x'')$ i postavi k na 1 ($k \leftarrow 1$); u suprotnom uvećaj k za 1 ($k \leftarrow k + 1$);
until $k > k_{max}$
until Kriterijumi za prekid su ispunjeni

Pseudokod pokazuje da promenljiva k, koja odredjuje okolinu u kojoj se izvršava razmrdavanje, uzima redom vrednosti od 1 do nekog k_{max}. To se može modifikovati tako što se uvode dva

parametra: k_{min} i k_{step}. Parametar k_{min} odredjuje minimalnu vrednost promenljive k (znači, umesto koraka $k \leftarrow 1$ stoji korak $k \leftarrow k_{min}$), dok parametar k_{step} odredjuje za koliko se uvećava promenljiva k, ako, nakon razmrdavanja i lokalne pretrage, nije pronadjeno bolje rešenje (što znači da će umesto $k \leftarrow k+1$ biti $k \leftarrow k+k_{step}$). Takodje, može se promeniti smer pretrage, tako da se prvo vrši razmrdavanje u većim okolinama (veća vrednost promenljive k), a zatim u manjim okolinama (vrednost promenljive k se smanjuje).

4.3 Metoda promenljivih okolina sa dekompozicijom

U okviru Metode promenljivih okolina sa dekompozicijom (engl. Variable neighborhood decomposition search, VNDS) [110] se primenjuje razbijanje problema na manje probleme (potprobleme) koji će se lakše rešiti. Kombinovanjem rešenja tih malih problema, dobija se rešenje polaznog problema. Pojedini koraci Metode promenljivih okolina daju osnove (mehanizme) za razbijanje (dekompoziciju) polaznog problema. Ako je x trenutno rešenje, a x' rešenje dobijeno razmrdavanjem u okolini $N_k(x)$, tada je moguće izdvojiti k atributa koji karakterišu rešenje x', ali ne karakterišu rešenje x. Preostali atributi rešenja su identični. Sa y se označava skup atributa rešenja x' koji karakterišu x', ali ne karakterišu x ($y = x' \setminus x$). Sužava se lokalno pretraživanje od rešenja x' kroz prostor rešenja uz pomoć dozvoljavanja samo promene atributa iz skupa y i zabranjivanja izmene atributa iz skupa $x \cap x' = x' \setminus y$. To se može tretirati i kao lokalno pretraživanje u prostoru mogućih rešenja za skup atributa y. Ako se u lokalnom pretraživanju popravi trenutno rešenje u y i dobije novo rešenje u y', onda taj y' i skup atributa $x' \setminus y$ odredjuju jedno novo rešenje x'' koje, po prirodi stvari, treba da bude bolje od rešenja x. Svi ostali koraci Metode promenljivih okolina sa dekompozicijom odgovaraju koracima VNS-a. To znači da, ako lokalnim pretraživanjem nije popravjeno rešenje za skup atributa y, postavlja se $k \leftarrow k+1$ (ili $k \leftarrow k+k_{step}$), a u suprotnom postavlja se $k \leftarrow 1$ (ili $k \leftarrow k_{min}$). Promenljiva k odredjuje okolinu u kojoj će se obaviti razmrdavanje trenutnog rešenja x, a kao što je primećeno, razmrdavanje odreduje manji problem koji se obradjuje. Time što lokalno pretraživanje sužava samo na prostor mogućih rešenja u y (koji može biti manji od prostora rešenja kompletnog problema), skraćuje se i njegovo trajanje, a time i trajanje celog postupka. Umesto lokalnog pretraživanja u prostoru mogućih rešenja po y, može se primeniti osnovna Metoda promenljivih okolina. Time se povećavaju šanse da se pronadje rešenje u y' koje je bolje od rešenja u y (vidi Algoritam 6).

Algoritam 6 Metoda promenljivih okolina sa dekompozicijom

1. Inicijalizacija. Odredi adekvatan skup okolina $N_k, k = 1, 2, ..., k_{max}$ odredi početno rešenje $x \in X$;
Izaberi kriterijum za prekid izračunavanja
repeat
a) Postavi k na 1 ($k \leftarrow 1$);

repeat
- Razmrdavanje: Izaberi na slučajan način rešenje x' iz $k$te okoline rešenja x ($x' \in N_k(x)$); odredi skup y atributa rešenja x' koji ne karakterišu rešenje x ($y = x' \setminus x$);
- Lokalno pretraživanje. Primeni neko lokalno pretraživanje po y polazeći od rešenja x'; Označi sa y' tako dobijeni lokalni optimum, x'' je odgovarajuće rešenje za polazni problem ($x'' = (x' \setminus y) \cup y'$);
- Da li se pomeriti? Ako je x'' bolje od x pomeri se u $x''(x \leftarrow x'')$ i postavi k na 1 ($k \leftarrow 1$);
U suprotnom uvećaj k za 1 ($k \leftarrow k+1$);
until $k > k_{max}$
until Kriterijumi za prekid nisu ispunjeni

4.4 Adaptivna metoda promenljivih okolina

Ukoliko je slučaj takav da je lokalni optimum daleko od ostatka prostora rešenja (ostalih lokalnih rešenja) i da se takav optimum nalazi izmedju dolina, govori se o uvrtanju okolina i velikim skokovima kroz prostor rešenja koji čine adaptivnu metodu promenljivih okolina (engl. Skewed variable neighborhood search, SVNS) [110]. Testiranja su pokazala da razmrdavanje i lokalna pretraga ne dovode nigde u tim slučajevima, odnosno do boljih lokalnih optimuma, a razlog su velika rastojanja, tako da dolazi do zarobljavana u tom udaljenom lokalnom optimumu i vrtenja oko njega. S druge strane, klasična metoda promenljivih okolina dozvoljava pomeranje iz jednog lokalnog optimuma samo u neki bolji lokalni optimum. Varijanta SVNS je kreirana da bi se vršilo efikasno pomeranje iz tog lokalnog optimuma, pa je predloženo da se dozvoli i skok u neki lošiji lokalni optimum dobijen nakon razmrdavanja i lokalne pretrage. Predlog metode se sastoji od prelazka iz lokalnog optimuma x u nešto lošiji lokalni optimum x'' ako se x'' "znatno" razlikuje od x. Prilikom odlučivanja da li će se vršiti pomeranje iz x u x'', koriste se vrednosti funkcije cilja za data dva rešenja, ali i za procenu koliko se ta dva rešenja razlikuju. Kao mera različitosti dva rešenja obično se uzima rastojanje izmedju njih i za to se koristi ista metrika koja je iskorišćena za definisanje skupa okolina. Dakle, kao kriterijum za prelazak iz jednog lokalnog optimuma u drugi, uzima se kombinacija vrednosti funkcije cilja i rastojanja izmedju lokalnih optimuma (vidi Algoritam 7). Sa x_{opt} označavamo najbolje rešenje tokom dosadašnje pretrage a sa f_{opt} vrednost funkcije cilja za to rešenje. U koraku tri proveravamo da li je rešenje dobijeno nakon lokalne pretrage x'' bolje od dosadašnjeg x_{opt} i ako jeste to postaje novo x_{opt}. U koraku četiri proveravamo da li ćemo se pomeriti u rešenje dobijeno nakon lokalne pretrage. Kao što smo rekli, pomericemo se čak i u slučaju da je to rešenje lošije, ako je dovoljno daleko. Primetimo da ako je $f(x'') < f(x_{opt})$ to će sigurno biti ispunjen i uslov četiri pa ćemo se svakako pomeriti. Napomenimo da se tokom pretrage formiraju dve trajektorije. Jednu čine uzastopne vrednosti x_{opt} a drugu uzastopne vrednosti trenutnog rešenja x. Za rešenja na prvoj trajektoriji važi da je svako sledeće bolje od prethodnog, a za drugu to ne mora važiti.

Algoritam 7 Adaptivna metoda promenljivih okolina

Inicijalizacija. Odredi adekvatan skup okolina $N_k, k = 1, 2, ..., k_{max}$;
Odredi početno rešenje $x \in X$, kao i vrednost funkcije cilja $f(x)$;
Postavi $x_{opt} \leftarrow x$ i $f_{opt} \leftarrow f(x)$;
Izaberi kriterijum za prekid izračunavanja
repeat
  a) Postavi k na 1 ($k \leftarrow 1$);
  repeat
    i)Razmrdavanje. Izaberi na slučajan način rešenje x' iz $k$te okoline rešenja x ($x' \in N_k(x)$);
    ii)Lokalno pretraživanje. Primeni neko lokalno pretraživanje od rešenja x'; Neka je x'' tako dobijeni lokalni optimum;
    iii)Da li ima popravke? Ako je $f(x'') < f_{opt}$, postavi $f_{opt} \leftarrow f(x'')$ i $x_{opt} \leftarrow x''$;
    iv)Da li se pomeriti? Ako je $f(x'') - \alpha\rho(x'', x) < f(x)$, pomeri se u njega ($x \leftarrow x''$) i produži pretraživanje od okoline $N_1(k \leftarrow 1)$;
    U suprotnom produži pretragu u narednoj okolini, tj. uvećaj k za 1 ($k \leftarrow k + 1$)
  until $k > k_{max}$
until Kriterijumi za prekid nisu ispunjeni

4.5 Redukovana metoda promenljivih okolina

Redukovana metoda promenljivih okolina (engl. Reduced variable neighborhood search, RVNS) [110] se dobija kada se izbaci lokalno pretraživanje. Odmah nakon razmrdavanja se proverava dobijeno rešenje i vrši se pomeraj u to rešenje ako je bolje od trenutnog. Ovim se skraćuje vreme pretrage u toku procedure jer lokalno pretraživanje ipak zahteva delimično ili kompletno istraživanje okoline N_1, što može potrajati. Nekad se RVNS koristi da se od početnog rešenja

koje može biti i lošije po karakteristikama stigne brzo do nekog boljeg rešenja, da bi se od tog boljeg rešenja primenila neka kvalitetnija pretraga.

Moguće je da se umesto jednog rešenja iz okoline $N_k(x)$, u koraku razmrdavanja, izabere nekoliko (m) (gde je m neki parametar) čime se povećavaju šanse za izbor pravog rešenja iz okoline gde postoji, ako postoji. Primenjivanje RVNS ima smisla sve dok u kolekciji okolina N_1, N_2, ..., $N_{k_{max}}$ ima relativno dosta rešenja boljih od postojećeg tako da se najobičnijim razmrdavanjem (izvedenim na slučajan način) može doći do boljeg od postojećeg rešenja (vidi Algoritam 8).

Algoritam 8 Redukovana metoda promenljivih okolina

Inicijalizacija. Odredi adekvatan skup okolina $N_k, k = 1, 2, ..., k_{max}$; Odredi početno rešenje $x \in X$; izaberi kriterijum za prekid izračunavanja
repeat
  a)Postavi k na 1 ($k \leftarrow 1$);
  repeat
    i)Razmrdavanje. Izaberi na slučajan način rešenje iz $k$te okoline rešenja x ($x' \in N_k(x)$);
    ii)Da li se pomeriti? Ako je $f(x') < f(x)$, pomeri se u njega ($x \leftarrow x'$) i produži pretraživanje od okoline N_1 ($k \leftarrow 1$);
    U suprotnom produži sa pretragom u narednoj okolini, tj. uvećaj k za 1 ($k \leftarrow k + 1$)
  until $k > k_{max}$
until Kriterijumi za prekid nisu ispunjeni

4.6 Generalna metoda promenljivih okolina

Ukoliko se Metoda lokalne pretrage zameni sa kompletnom Metodom spusta kroz promenljive okoline (VND) u okviru metode VNS-a, dobija se Generalna metoda promenljivih okolina (engl. General variable neighborhood search, GVNS). U odnosu na Osnovnu metodu, ova ekstenzija je dovela do najuspešnijih rezultata u primeni. U okviru Osnovne metode VNS koriste se okoline koje su indukovane iz jedinstvene metrike, dok se u okviru GVNS metode, baš zbog korišćenja VND metode umesto lokalne pretrage, okoline mogu indukovati iz različitih metrika i mogu se razmatrati simultano (vidi Algoritam 9).

Algoritam 9 Generalna metoda promenljivih okolina

repeat
  $k \leftarrow 1$;
  repeat
    $x' \leftarrow Razmrdavanje(x, k)$;
    $x'' \leftarrow VND(x', k'_{max})$;
    PromenaOkoline(x, x'', k);
  until $k = k_{max}$;
until Kriterijumi za prekid nisu ispunjeni

4.7 Problem raspoređivanja zadataka

U ovom odeljku predstavljena su dva rada u kojima je rešavan problem rasporedjivanja. U njima je prikazana primena i testiranje Heurističkih algoritama Monte Carlo i VNS.
U računarstvu, rasporedjivanje (engl. Scheduling) je problem u kome se posao deli, odnosno dodeljuje, na više resursa. Resursi mogu biti kako hardverski elementi kao što su procesori, memorija, mreže, tako i virtuelni računarski elementi, kao npr. male sekvence instrukcija.

4.7.1 Heuristički pristup rasporedjivanju nezavisnih zadataka na identične procesore

Razmatran je poznat problem rasporedjivanja n nezavisnih zadataka na homogene multiprocesorske sisteme koji sadrže p identičnih procesora [69]. Iako je ovo najjednostavnija varijanta problema rasporedjivanja, dokazano je da je to NP-kompletan problem u strogom smislu [37]. U literaturi postoji nekoliko egzaktnih metoda koje rešavaju probleme male i srednje veličine, a takodje i izvestan broj (meta)heurističkih metoda [202]. Kod većine algoritama, redosled dodeljivanja zadataka odredjuje se na osnovu prioriteta koji zavisi od dužine (vremena) njihovog izvršavanja.
Prikazan je Heuristički algoritam koji se koristi za rešavanje velikih primera razmatranog problema. Monte Karlo je najjednostavnija metaheuristička procedura pretrage koja se bazira na slučajnom pretraživanju kroz prostor rešenja. Kompleksnije metaheurističke metode kao što su Genetski algoritmi, Tabu pretraga, Metoda promenljivih okolina, Neuronske mreže su dizajnirane za sistematičniju pretragu kroz prostor rešenja, ali zahtevaju i sofisticiranije strategije za implementaciju.
Na osnovu računarski testova i rezultata, Monte Karlo se izvršava dobro na instancama koje su srednje veličine.

Formulacija problema

Problem rasporedjivanja nezavisnih zadataka na homogene multiprocesorske sisteme se sastoji od sledećeg: Neka je dat skup zadataka $T = \{1, 2, ..., n\}$ i skup identičnih procesora $P = \{1, 2, ..., p\}$. Pretpostavlja se da je n broj zadataka, a da je p broj procesora i da su to ulazni parametri. Za sve zadatke, dato je procesorsko vreme $l_i, i = 1, ..., n$. Cilj procesa rasporedjivanja je da pridruži sve zadatke procesorima na taj način da minimizuje ukupno vreme završetka svih zadataka koje naziva se dužina raspodele (engl. Makespan). Svaki zadatak treba da se izvrši na jednom procesoru i svaki procesor izvršava samo jedan zadatak u trenutku.
Dakle, vreme izvršavanja svakog procesora je odredjeno sumom procesorskih vremena svih zadataka koji su pridruženi tom procesoru. Dužina raspodele je jednaka maksimumu po svim procesorskim vremenima izvršavanja. Preciznije, ako se definiše

$$T_j = \{i : \text{zadatak } i \text{ se dodeljuje procesoru } j\}$$

tada se dužina raspodele može izračunati kao:

$$y = C_{max} = \max_{1 \le j \le p} C_j$$

gde je C_j vreme rada procesora j i to je ukupno vreme za koje se završi izvršavanje svih zadataka koji su dodeljeni tom procesoru:

$$C_j = \sum_{i \in T_j} l_i$$

Formulacija matematičkog programiranja se uvodi u odnosu na sledeću definiciju promenljivih

$$x_{ij} = \begin{cases} 1, & \text{ako je zadatak } i \text{ pridružen procesoru j} \\ 0, & \text{inače} \end{cases}$$

gde je $i = 1, \ldots, n$, a $j = 1, ..., m$.

Tada je formulacija mešovitog celobrojnog programiranja (engl. Mixed integer programming, MIP) problema rasporedjivanja data sa:

$$\min y \tag{4.1}$$

p.o.

$$\sum_{j=1}^{p} x_{ij} = 1, 1 \le i \le n \tag{4.2}$$

$$y - \sum_{i=1}^{n} l_i x_{ij} \ge 0, 1 \le j \le p \tag{4.3}$$

$$x_{ij} \in \{0,1\}, 1 \leq i \leq n, 1 \leq j \leq p \tag{4.4}$$

Cilj je minimizirati dužinu raspodele y tako da budu zadovoljena zadata ograničenja. Skup ograničenja 4.2 označava da svaki zadatak može biti pridružen tačno jednom procesoru. Dužina raspodele se računa kao maksimum svih vremena izvršavanja procesora, što je opisano ograničenjima 4.3. Ograničenja 4.4 pokazuju da su x_{ij} binarne promenljive. Ovakva formulacija predstavlja jednostavan linearan program i koristi se za pronalazak optimalnog rešenja za test primere manjih dimenzija. Za implementaciju su korišćeni softverski paketi ILOG AMPL i CPLEX 11.2.

Heurističko rešenje

Početno rešenje dobijeno je primenom konstruktivne heuristike koja gradi rešenje na osnovu datih ulaznih podataka. To rešenje se zatim poboljšava primenom jednostavnih iterativnih algoritama slučajne pretrage. Konstruktivna heuristika sastoji se iz dva koraka: prvo se pravi lista zadataka preko slučajnog uredjivanja, a zatim se zadaci pridružuju najmanje opterećenom procesoru. Implementirana je MC metoda, koja se sastoji od iterativnog slučajnog rasporedjivanja izvršavanja i od verifikacije kvaliteta rešenja sve dok se uslov zaustavljanja ne ispuni. Najbolji dobijeni rezultat se zapisuje kao konačan. Uslovi zaustavljanja mogu biti: maksimalan broj iteracija, maksimalan broj iteracija bez poboljšanja, maksimalno dozvoljeno CPU vreme, itd.

Eksperimentalni rezultati

Heuristička implementacija se testira na slučajnim instancama koje su generisane preko procedura opisanih u [68]. Koriste se primeri malih i srednjih dimenzija sa poznatim optimalnim rešenjem. Optimalno rešenje je dobijeno i preko ILOG AMPL i preko CPLEX 11.2 softverskih paketa. Program se izvršavao na Intel Core 2 Duo CPU E6750 na 2.66GHz sa RAM=8 Gb pod Linux Slackware 12, Kernel: 2.6.21.5, gcc verzija 4.1.2. Eksperimentalni rezultati za testove malih i srednjih dimenzija (50 i 100 zadataka) su sumirani u Tabeli 4.1. Prva kolona Tabele 4.1 sadrži ime primera. Broj zadataka je deo imena fajla, dok je broj procesora dat u drugoj koloni. U trećoj i četvrtoj koloni prezentovana je dužina optimalne raspodele i CPU vreme koje je CPLEX-u bilo potrebno, respektivno. Kolona pet sadrži dužinu raspodele koja je postignuta Monte Karlo metodom, dok je odgovarajuće CPU vreme (potrebno Monte Karlo metodi da nadje najbolje rešenje) dato u šestoj koloni. Poslednja kolona sadrži ukupno vreme izvršavanja Monte Karlo metode (dok se kriterijum zaustavljanja ne ispuni). Postavljen je kriterijum zaustavljanja na 10000 iteracija. Testni primeri su podeljeni u dve grupe, prva sadrži slučajne primere koji su laki za rasporedjivanje. Optimalno rešenje se dobija preko Monte Karlo metode za sve instance sem jedne. Sa druge strane, primeri za koje je potimalna raspodela fiksirana tokom procesa generisanja su teški za rešavanje Monte Karlo metodom, kao i CPLEX paketom. Vreme izvršavanja CPLEX paketom raste ekstremno sa porastom broja procesora p. MEMERROR u Tabeli 4.1 za $Iogra100\_00\_12$ primer znači da nijedan rezultat nije dobijen CPLEX paketom zbog nedostatka memorije. Takodje, nije rešeno ni $Iogra100\_00\_16$. Za ove i dva primera sa $n = 50$ zadatka nije ni gubljeno vreme na izvršavanje CPLEX-a jer je optimalno rešenje poznato. Kao što se vidi u Tabeli 4.1, Monte Karlo generiše kvalitetna rešenja (odstupanje od optimuma je manje od 7% u najgorem slučaju) za veoma kratko vreme izvršavanja. Ukupno vreme rada za 10000 iteracija je manje od 1 sekunde za sve primere.

Tabela 4.2 prikazuje poboljšane rezultate, za probleme koji ranije nisu rešeni, kod kojih je kriterijum zaustavljanja udesetostručen. Rešavani su samo primeri sa poznatim optimalnim rešenjem. Kad se uporede stara i nova rešenja primećuje se malo poboljšanje u kvalitetu rezultata, dok je vreme izvršavanja poraslo 10 puta. Time je potvrdjeno da je Monte Karlo suviše jednostavna metoda, slučajna pretraga koja nema karakteristiku sistematičnog poboljšanja. Zato je očito da treba primeniti neke sofisticiranije metaheuristike kao što su GA, VNS i TS kako bi se dobila još kvalitetnija rešenja. U narednom odeljku je opisana implementacija VNS metode.

4.7.2 VNS za problem rasporedjivanja zadataka

Predlozena implementacija VNS metode za problem rasporedjivanja opisana je u [70]. Algoritam VNS je uspešno primenjen na razne varijante kombinatornih i kontinualnih problema optimizacije, na primer na Problem multiprocesorskog rasporedjivanja sa komunikacionim kašnjenjem [66]. Implementacija metode se zasniva na dinamičkim strukturama podataka koje obezbedjuju efikasno ažuriranje funkcije cilja. Najznačajnija struktura je dinamička matrica $S_{p\times n}$ čiji element s_{ji} sadrži indeks i-tog zadatka koji je rasporedjen na procesor j. Za svaki procesor,

Tabela 4.1: Rezultati rasporedjivanja - medium problemi

example	p	OPT	Opt CPU time	Monte Karlo	min CPU time	CPU time
It50_70	4	212	0.020996	212	0.006998	0.044993
It50_80	4	196	0.027996	196	0.001000	0.041994
It50_80_1	4	234	0.043994	234	0.001000	0.040994
It50_80_2	4	337	0.019997	337	0.006999	0.040994
It50_80_3	4	216	0.025996	216	0.000000	0.041994
It50_80_4	4	276	0.017996	276	0.028996	0.041994
It50_80_5	4	128	0.006999	128	0.000000	0.042994
It50_80_6	4	167	0.022997	167	0.001000	0.042994
It100_50_1	4	471	0.097985	471	0.001000	0.0080988
It100_60_1	4	465	0.009999	465	0.001000	0.084987
It100_40_1	4	493	0.063991	493	0.005999	0.081987
It100_40_2	4	782	0.110983	783	0.019997	0.080988
It100_40_3	4	478	0.095986	478	0.027996	0.079988
It100_40_4	4	483	0.074989	483	0.009999	0.083988
It100_40_5	4	271	0.035995	271	0.000000	0.080988
It100_40_6	4	340	0.010999	340	0.046993	0.079988
Iogra50_00_2	2	600	0.010000	600	0.000000	0.032994
Iogra50_00_4	4	600	0.269959	**601**	0.000000	0.040994
Iogra50_00_6	6	600	71.234200	**605**	0.011999	0.048993
Iogra50_00_8	8	600	8304.470000	**611**	0.030996	0.056992
Iogra50_00_9	9	600	24751.800000	**617**	0.037994	0.057991
Iogra50_00_12	12	600	-	**631**	0.022997	0.062991
Iogra50_00_16	16	600	-	**641**	0.001000	0.069990
Iogra100_00_2	2	800	0.002999	800	0.000999	0.067989
Iogra100_00_4	4	800	0.139978	**801**	0.001000	0.081988
Iogra100_00_6	6	800	8.401720	**803**	0.081988	0.094986
Iogra100_00_8	8	800	98.859000	**807**	0.073989	0.108984
Iogra100_00_9	9	800	862.788000	**809**	0.042994	0.111983
Iogra100_00_12	12	800	MEMERROR	**821**	0.043994	0.121982
Iogra100_00_16	16	800	-	**829**	0.013998	0.135980

pridružena vrsta matrice S sadrži listu zadataka koji su rasporedjeni na odgovarajući procesor. Redosled zadataka na svakom od procesora u principu nije bitan, ali se ovde uzima u obzir zbog lakše manipulacije strukturama za predstavljanje rešenja. Kako nisu svi elementi matrice S relevantni, uveden je niz $o_j, j = 1, \ldots, p$ koji sadrži brojeve zadataka pridruženih procesoru j. Poslednji deo rešenja jeste niz $y_j, j = 1, \ldots, p$ čiji elementi predstavljaju vreme rada odgovarajućih procesora, tako da je y_j jednako sumi vremena izvršavanja svih zadataka koji su pridruženi procesoru j.

Koriste se dva tipa okolina: pomeranje (engl. Shift) i zamena (engl. Interchange). Shift okolina se realizuje pomeranjem zadataka od jednog procesora do drugog. To znači da kompleksnost shift okoline iznosi $O(np)$, odnosno da se svaki zadatak, a ima ih n, premešta na neki od preostala $p - 1$ procesora. Intechange okolina znači da zadaci sa različitih procesora razmenjuju svoje pozicije. Najgori slučaj kompleksnosti interchange okoline jeste $O(n^2)$ s obzirom da uzima u obzir za promenu pozicije svaki par zadatka koji nije rasporedjen na isti procesor.

Kako bi se kreirala efikasnija implementacija, korišćene su redukovane okoline. Shift okolina pomera samo zadatke koji su na najopterećenijem procesoru na najmanje opterećen procesor. U okviru interchange okoline samo zadaci sa najopterećenijeg procesora i najmanje opterećenog procesora menjaju svoje pozicije. Kako redosled zadataka nije relevantan u slučaju rasporedjivanja nezavisnih zadataka, pomereni zadaci se smeštaju na kraj liste datog procesora. Prazan prostor koji je zadatak koji je pomeren ostavio iza sebe se popunjava sa poslednjim zadatkom koji je alociran za taj procesor. Ovakve pretpostavke olakšavaju ažuriranje strukture podataka u svakom koraku lokalne pretrage. Za svaku okolinu broj transformacija jednog rešenja u drugo je konstanta, tj. kompleksnost pomeraja iz jednog rešenja u drugo jeste $O(1)$. Na primer, kod

Tabela 4.2: Poboljšani rezultati rasporedjivanja - medium problemi

example	p	OPT	Opt CPU time	Monte Karlo	min CPU time	CPU time
It100_40_2	4	782	0.110983	782	0.389940	0.839872
Iogra50_00_4	4	600	0.269959	600	0.194971	0.413937
Iogra50_00_6	6	600	71.234200	**604**	0.025996	0.502924
Iogra50_00_8	8	600	8304.470000	**610**	0.043994	0.552916
Iogra50_00_9	9	600	24751.800000	**613**	0.076988	0.595910
Iogra50_00_12	12	600	—	**617**	0.076989	0.638903
Iogra50_00_16	16	600	—	**640**	0.377943	0.696894
Iogra100_00_4	4	800	0.139978	800	0.540917	0.821875
Iogra100_00_6	6	800	8.401720	**802**	0.091986	0.964854
Iogra100_00_8	8	800	98.859000	**806**	0.055992	1.119830
Iogra100_00_9	9	800	862.788000	**807**	0.798879	1.144826
Iogra100_00_12	12	800	MEM_ERROR	**817**	0.743887	1.221815
Iogra100_00_16	16	800	—	**824**	0.715892	1.356794

okoline shift, za pomeraj zadatka i (koji se nalazi kao r-ti a procesoru j) iz procesora j u procesor m potrebno je izvesti sledeće korake:

- $s[m][o[m]++] = i$
- $s[j][r] = s[j][--o[j]]$
- $y[m] += l[i]$
- $y[j] -= l[i]$

Eksperimentalni rezultati

Heuristička implementacija je testirana na primerima malih dimenzija sa poznatim optimalnim rešenjima. Eksperimenti su vršeni u istim uslovima kao i u prethodnom slučaju. Uporedjivano je vreme izvršavanja sa kvalitetom rešenja za ove programe. Test primeri su generisani koristeći slučajno generisanje predloženo u [68]. Implementirana je osnovna VNS metoda. To znači da su okoline testirane odvojeno i kao bolja se pokazala okolina sa pomeranjem zadataka sa jednog procesora na drugi Eksperimentalni rezultati su sumirani u Tabeli 4.3. Prva kolona Tabele 4.3 sadrži naziv instance. Broj zadataka je deo imena, dok je broj procesora postavljen na četiri u svim primerima. U drugoj i trećoj koloni predstavljena je dužina optimalne respodele i CPU vreme koje je potrebno CPLEX-u za njeno pronalaženje. Kolona četiri sadrži dužinu raspodele koju je dobio VNS. Pridruženo CPU vreme koje je potrebno da VNS pronadje najbolje rešenje se prikazuje u poslednjoj koloni.

Kako se može videti iz Tabele 4.3, VNS je sposoban da pronadje optimalno rešenje za sve test instance sa značajno manjim CPU vremenom. Kriterijum zaustavljanja za VNS je postavljen na 1 sec CPU vremena. Uporedjivano je izvršavanje VNS heuristike sa prethodno implementiranim Monte Karlo algoritmom [69]. Kriterijum zaustavljanja za Monte Karlo je postavljen na 10,000 iteracija i trebalo je manje od 1 sec da se proces završi. Rezultati su predstavljeni Tabelom 4.4.

Iako je Monte Karlo sposoban da pronadje optimalno rešenje za najveći broj instanci, potrebno mu je nekoliko puta duže vreme izvršavanja nego VNS algoritmu u većini slučajeva.

Tabela 4.3: Rezultati rasporedjivanja za male primere

Example	opt	opt CPU time	VNS	VNS CPU time
It50_70	212	0.021	212	**0.001**
It50_80	196	0.028	196	**0.001**
It50_80_1	234	0.044	234	**0.001**
It50_80_2	337	0.020	337	**0.001**
It50_80_3	216	0.026	216	**0.001**
It50_80_4	278	0.018	278	**0.001**
It50_80_5	128	0.007	128	**0.001**
It50_80_6	167	0.023	167	**0.001**
It100_40_1	493	0.064	493	**0.001**
It100_40_2	782	0.111	782	**0.001**
It100_40_3	478	0.096	478	**0.001**
It100_40_4	483	0.075	483	**0.001**
It100_40_5	271	0.036	271	**0.001**
It100_40_6	340	0.011	340	**0.001**
It100_50_1	471	0.098	471	**0.001**
It100_60_1	465	0.010	465	**0.001**

Tabela 4.4: Uporedjivanje VNS i Monte Karlo

Example	Monte Karlo	Monte Karlo CPU time	VNS	VNS CPU time
It50_70	212	0.007	212	**0.001**
It50_80	196	0.001	196	0.001
It50_80_1	234	0.001	234	0.001
It50_80_2	337	0.007	337	**0.001**
It50_80_3	216	0.000	216	0.001
It50_80_4	278	0.029	278	**0.001**
It50_80_5	128	0.000	128	0.001
It50_80_6	167	0.001	167	0.001
It100_40_1	493	0.006	493	**0.001**
It100_40_2	783	0.020	782	**0.001**
It100_40_3	478	0.028	478	**0.001**
It100_40_4	483	0.001	483	0.001
It100_40_5	271	0.000	271	0.001
It100_40_6	340	0.047	340	**0.001**
It100_50_1	471	0.001	471	0.001
It100_60_1	465	0.001	465	0.001

4.8 Metoda promenljivih formulacija

Metoda promenljivih formulacija ili Metoda pretrage prostora formulacija (engl. Formulation Space Search, FSS) je jedna od novijih ekstenzija Metode promenljivih okolina i suštinski je najkompleksnija metoda. Ovaj algoritam pored promena okolina i metrike sadrži i promenu formulacija, odnosno korišćenje različitih definicija za problem koji se rešava. Takodje, u zavisnosti od karakteristike modela, zavisi i implementacija FSS metode. Algoritam VNS u svom izvornom obliku sadrži korak promene okolina, ali u slučaju FSS metode nije neophodno da se izvodi promena okoline ukoliko se promenom formulacija dobija kvalitetno rešenje. Sa druge strane, nije loše ispitati promene formulacija paralelno sa promenama okolina i uporediti dobijena rešenja. Cilj je dobiti efikasnije rešenje, samo što implementacija promena formulacija može biti komplikovanija i zahtevnija nego implementacija osnovne VNS metode. Svaka formulacija ima svoju sopstvenu metriku u okviru lokalne pretrage, a i svoje sopstveno razmrdavanje rešenja. Struktura okolina je drugačija i jedino što ih povezuje jeste funkcija transformacije koja transformiše rešenja preko rastojanja izmedju formulacija.
Neka je $\mathcal{F}$ prostor formulacija i on predstavlja skup različitih formulacija istog problema. Kako

bi se obezbedila pretraga kroz prostor formulacija potrebno je definisati distancu, koja opisuje okoline $\{\mathcal{N}_l(\Phi)|l = 1, ..., l_{\max}\}$ formulacije $\Phi \in \mathcal{F}$. Svaka formulacija jeste definicija problema koja se zapisuje pomoću nekog oblika matematičkog programiranja, odnosno, preko funkcije cilja i mogućih odredjenih ograničenja. Kako bi se pronašao globalni optimum problema potrebno je uporediti lokalne optimume različitih formulacija, odnosno vrednosti funkcije cilja, i analizirati koja od njih jeste približna globalnom rešenju. Kako bi se pronašle bolje formulacije u trenutnom rešenju pretraga se vrši kroz $\mathcal{F}$. Ideja Metode pretrage kroz prostor formulacija u okviru VNS-a je nastala na osnovu toga što se većina problema može formulisati na više načina. Ona se svodi na promene formulacija u okviru lokalne pretrage, razmrdavanja i promena okolina u bazičnom VNS-u. Odredjivanje distance izmedju bilo koje dve formulacije se vrši za svaki problem pojedinačno i ne može se utvrditi univerzalno pravilo za to. Na taj način se zamke lokalnih optimuma izbegavaju tako što se pretraga vrši ne samo kroz prostor rešenja već i kroz prostor formulacija. Algoritam FSS je prvi put predložen za problem pakovanja krugova. U radu o pakovanju krugova (engl. Circle Packing Problem, CPP) [171] pokazano je da stacionarna tačka nelinearne formulacije u dekartovim koordinatama nije i stacionarna tačka u polaranom koordinatnom sistemu. Tu je uvedena metoda Reformulation descent (RD) koja alternira izmedju ove dve formulacije do pronalaženja konačnog rešenja. Nakon svake lokalne pretrage sa kranjim rešenjem x, nova lokalna pretraga započinje od trenutnog rešenja x, transformisanog i zapisanog u drugim koordinatama.
Ipak, sama ideja FSS omogućava više od dve formulacije. Pitanje izbora formulacije je takodje predmet razmatranja. Postoji mogućnost da definišemo samo jednu formulaciju i taj slučaj je jednostavan za rešavanje. Ukoliko postoje dve linearno povezane formulacije, petlja algoritma postaje trivijalna, i imaće istu stacionarnu tačku, što ne dovodi do efikasnosti u pretrazi rešenja. Ukoliko se desi slučaj da stacionarna tačka jedne formulacije bude lokalni optimum druge formulacije takodje se dobija neefikasna procedura.

4.8.1 Opšti slučaj FSS metode

Algoritam 10 Metoda promenljivih formulacija (bez promena okolina)

Definisati prostor formulacija $\mathcal{F}$ i definisati metriku u prostoru formulacija $N_k(\phi)$ gde je $k = 1, ..., k_{max}$
Neka su (ϕ, x) početna formulacija i početno rešenje
Neka je f funkcija cilja u formulaciji ϕ
Primeniti Metodu lokalne pretrage u ϕ sa početnom tačkom x i dobiti x^*, $x^* \leftarrow Lokalnapretraga(\phi, x)$
$k \leftarrow 1$
repeat
  $\phi' \leftarrow Slucajnaformulacija(N_k(\phi))$
  $x' \leftarrow Transformacija(\phi', x)$
  $x'' \leftarrow Lokalnapretraga(\phi', x')$
  if $f(\phi', x'') < f(\phi, x^*)$ **then**
    $(\phi, x^*) \leftarrow (\phi', x'')$
    $k \leftarrow 1$
  else
    $k \leftarrow k + 1$
  end if
  if $k > k_{\max}$ **then**
    $k \leftarrow 1$
  end if
until
Uslovi zaustavljanja su ispunjeni

Generalno, formulacije se koriste u slučajevima kada postoji dosta kvalitetnih lokalnih optimuma u okolini u odnosu na prethodne formulacije koje su korišćene. Bilo koje rešenje koje se pronadje u jednoj formulaciji moguće je lako prevesti u neku drugu formulaciju. Prelazi se

Algoritam 11 Lokalna pretraga - detaljno

```
Poboljsanje ← false
repeat
    for y ∈ N(x) do
        if f(φ, y) < f(φ, x) then
            x ← y
            Poboljsanje ← true
            Break
        end if
    end for
until Poboljsanje
```

iz jedne formulacije u drugu tako što se iz početne formulacije nakon lokalne pretrage dobija inicijalno rešenje, koje se zatim koristi u sledećoj formulaciji (videti Algoritam 10). Tako se dolazi do finalnog rešenja. Ovakva strategija je korisna u slučaju kad se lokalne pretrage različitih formulacija ponašaju različito. Redosled izbora formulacija je važan i treba dobro proceniti koja formulacija prva započinje proces, a koja sledeća nastavlja, itd. Jedna od metoda je kreirati redosled formulacija tako da se odredi prelaz iz jedne formulacije u drugu tako da prednost imaju funkcije koje su linearno nezavisne. Svaki put kad se pronadje bolje rešenje, vrši se zadržavanje u njemu i nastavlja se pretraga iz formulacije u kojoj je pronadjeno to bolje rešenje. Druga mogućnost je da se rešenja svih ostalih formulacija proveravaju i uporedjuju dok se ne prekine proces.

5

Heuristike za problem maksimalne klike u grafu

U ovom poglavlju predstavljeni su algoritmi koji su implementirani i testirani a vezani su za problem koji se posmatra tokom istraživanja. Prvi algoritam koji je primenjen odnosio se na probem težinske klike. Nad njim je testirano nekoliko heuristika, a sada se predstavlja VNS rezultat [136]. Nakon toga, posmatran je problem particije klika i takodje razvijan VNS, kao i ekstenzije VNS [32]. Istraživanje je nadalje usmereno ka proučavanju nekonveksnih funkcija u globalnoj optimizaciji. Na osnovu toga, testiran je model maksimalne klike preko kontinualne funkcije na četiri algoritma preuzeta iz literature [75, 76, 130, 175]. Algoritmi su implementirani u skladu sa objavljenim radovima, a u skladu sa postojećim istraživanjem su adaptirani. Ispitivalo se da li bi kombinacija postojećih metoda sa VNS algoritmom dovela do boljih rezultata za nekonveksne kontinualne funkcije, a pored toga, ispitivalo se da li FSS algoritam u kombinaciji sa postojećim metodama daje bolje rezultate. Četiri kreirana algoritma adaptirana za nekonveksni kontinualni problem su dala solidne rezultate na malim skupovima grafova. Nakon toga, istraživanje je fokusirano na rad [195] i na osnovu njega su kreirana tri algoritma, tj. egzaktni algoritam, pod nazivom FEHO i dva heuristička algoritma, FHHO i FSSHO. FHHO je baziran na metodi selekcije čvora gde se posmatra stepen čvora višeg reda, a FSSHO je baziran na FSS metodi. Sve tri metode su dale značajne rezultate u odredjivanju kardinalnosti maksimuma klike što se može pogledati u [137–139].

5.1 Problem težinske klike

U ovom odeljku predstavljeni su rezultati primene VNS heuristike na problem maksimalne težinske klike (engl. Maximal vertex weighted clique, MVWC) u grafu. Dobijeni rezultati su bili ohrabrujući za nastavak istraživanja ove teme. Metoda je primenjena na testiranje velikih instanci, i uporediva sa drugim heuristikama kao na primer sa Genetskim algoritmima, Tabu pretragom ili Multistart lokalnom pretragom. Metoda je predstavila nove strukture okolina kombinujući ih u okviru VNS faza.
Za dati graf $G = (V, E)$ i funkciju W koja svakom čvoru grafa dodeljuje jedan integer koji se naziva težinom datog čvora, definiše se problem maksimalne težinke klike kao optimizacioni problem pronalaženja klike koja ima maksimalan zbir težina čvorova.
Razvijen je osnovni VNS algoritam koji se sastoji od tri faze u unutrašnjoj petlji: razmrdavanje, lokalna pretraga i promena okolina. Algoritam je poboljšan dodavanjem metode plato pretrage (engl. Plateau search).
Opisaće se teorijska osnova algoritma. Neka je dat neusmeren graf $G(V, E)$ sa $V = \{1, 2, ..., n\}$, $E \subseteq \{\{i, j\} : i, j \in V\}$, tada prostor rešenja Σ od MVWC može biti predstavljen kao skup svih kompletnih podgrafova od G ili ekvivalentno, svih stabilnih skupova od $\bar{G}$. Kako bi se konstruisale različite strukture okolina i kako bi se izvela sistematična pretraga potrebno je bilo pronaći distancu izmedju bilo koja dva rešenja, odnosno, potrebno je bilo da se obezbedi prostor sa nekom metrikom (ili kvazi metrikom) i nakon toga definišu okoline indukovane tom metrikom.

Definisane su distance $\rho(X, X')$ izmedju bilo koja dva rešenja X i X' kao kardinalnost simetrične razlike izmedju ovih skupova [112]:

$$\rho(X, X') = \frac{1}{2}|(X \cup X') \setminus (X \cap X')|$$

Neka je N_k, $(k = 1, ..., k_{max})$, konačan skup izabranih struktura okolina, a sa $N_k(X)$ označen skup rešenja u k-toj okolini od X. Sastoji se od svih rešenja na distanci k od X

$$N_k(X) = \{X' \in \Sigma, \rho(X, X') = k\}$$

Jasno je da je ova funkcija metrika, i da je Σ metrički prostor. $N_1(X)$ odgovara dodavanju ili brisanju čvorova u jednom momentu iz trenutnog rešenja. Odnosno, ${N_1}^-(X)$ odgovara brisanju jednog čvora a ${N_1}^+(X)$ dodavanju jednog čvora u trenutnu aproksimaciju:

$$N_1(X) = {N_1}^+(X) \cup {N_1}^-(X)$$

$${N_k}^+(X) \cup {N_k}^-(X) \subset N_k(X)$$

za $k \geq 2$ poslednji skupovi odgovaraju dodavanju ili brisanju k čvorova u momentu. Oznaka $\rho(.,.)$ se može smatrati Hamingovim rastojanjem, ako se rešenje X predstavi kao $0-1$ niz dužine n. Ako je X klika u G tada je $W(X)$ ukupna težina čvorova od X.

Neka je $C_p(X) = \{i \in V : |X \setminus N(i)| = p\}, p \in \{0, 1\}$ skup svih čvorova koji nisu susedni sa tačno p čvorova u X. Za uzastopne okoline važi da je $N_k(X) \subset N_{k+1}(X)$ odnosno da su ugnježdene [112], što znači da se koristi sekvencijalan poredak struktura okolina. Konačno rešenje je lokalni optimum u odnosu na sve okoline. Za pretragu prostora primenjen je algoritam GVNS. Inicijalno rešenje se dobija preko lokalne pretrage iz heuristike VND. Ista lokalna pretraga se koristi kasnije u okviru generalnog VNS. Skup od tri strukture okoline koje se koriste u GVNS su:
(i) brisanje (engl. drop);
(ii) dodavanje (engl. add);
(iii) zamena (engl. interchange)
koja se koristi kada se pokreti brisanja i dodavanja izvode u isto vreme. Čvor proizvoljnog grafa G je simplicijalni ako su svaka dva suseda čvora v medjusobno povezana u grafu G. Drugim rečima, čvor v je simplicijalan ako i samo ako je podgraf $G(N(v) \cup \{v\})$ indukovan skupom suseda čvorova v (zajedno sa čvorom v) kompletan graf. Simplicijalni čvor v ima veličinu (engl. size) l ako i samo ako ima tačno $l - 1$ suseda (odnosno klika koju čine taj čvor i njegovi susedi ima l elemenata). U [112] je pokazano da svaki čvor koji je simplicijalan u komplementarnom grafu G pripada bar jednoj maksimalnoj kliki grafa G. Ovo svojstvo je korišćeno za dva algoritma za odredjivanje maksimalne klike: u jednom algoritmu su razmatrani simplicijalni čvor čija veličina nije veća od $l_{max} = 2$, dok su u drugom razmatrani simplicijalni čvorovi čija veličina ne premašuje $l_{max} = 3$. U metodi promenljivih okolina koriste se simplicijalni čvorovi kao jedan deo VND pretraživanja. Naš VND koristi pokrete dodavanja u okviru faze spusta, a pokrete zamene ukoliko najbolje rešenje u okolini ima istu kardinalnost kao X, za plato fazu. Pokreti brisanja se koriste u fazi razmrdavanja VNS. Predstavljen je kratak opis koji je sličan kao u radu Hansen i drugi 2004 [112], i koji je primenjen na algoritam težinske klike. Neka je t iteracija u kojoj je, X_t rešenje trenutne aproksimacije, a V_t čvorovi grafa G_t koji ne formiraju ni maksimalnu kliku niti minimalnu transverzalu. E_t je podskup skupa ivica E kojima su oba kraja u V_t.
(i) pravilo pohlepne (engl. greedy) selekcije odredjuje kako odabrati sledeći čvor za dodavanje u X_t u okviru podproblema koji je definisan u grafu $G(V_t, E_t)$.
(ii) tie-breaking pravilo odredjuje da ako postoji više od jednog čvora koji zadovoljava selekciono pravilo gore, kako napraviti izbor u tome?
(iii) plato pretraga odredjuje da li pretraga treba da se nastavi u okolini čiji elementi imaju istu kardinalnost kao trenutno pronadjeno rešenje ($|X_{t+1}| = |X_t|, V_{t+1} = V_t = \emptyset$) i kako?
Kako su ova pitanja globalna za problem, opisan je kratak pregled strategije koja je primenjena. Čvorovi u grafu koji nisu u trenutnoj aproksimaciji MVWC su podeljeni na tri grupe: C_0 sadrži čvorove kandidate koji su povezani sa svim čvorovima u trenutnoj aproksimaciji MVWC, C_1 je grupa čvorova kandidata koji su povezani sa svima osim sa jednim čvorom u trenutnoj aproksimaciji MVWC i treća grupa su ne-kandidati. Ove liste se osvežavaju nakon svake promene u aproksimaciji MVWC. U fazi lokalne pretrage bira se jedan čvor iz C_0, ako postoji bilo koji

kandidat u C_0. Razvijena je funkciju prioriteta koja uključuje težine čvorova, stepen čvorova i penal na osnovu ukupnog izračunatog perioda čvorova u aproksimaciji klike:

$$h[x] = (\text{weight}[x] + \max_y \text{ time\_in\_clique}[y] \ - \ \text{time\_in\_clique}[x])\ \text{degree}[x]$$

Ovaj kandidat se dodaje u kliku i nakon toga se skupovi C_0 i C_1 ažuriraju. Sledeći korak je da se poveća ukupna težina klike izborom jednog C_1 kandidata koji se uključuje u trenutnu aproksimaciju maksimalne klike i izborom jednog kandidata iz trenutne aproksimacije maksimalne klike koji će iz nje biti izbačen. Ovo se naziva plato pretraga. Plato pretraga je procedura kojom se ne menja broj čvorova u trenutnoj aproksimaciji maksimalne težinske klike. Lokalna pretraga se završava kada je nemoguće pronaći čvorove kojima bi se povećala težina trenutne aproksimacije klike maksimalne težine prethodno opisanom transformacijom.

Lokalna pretraga alternira izmedju faze iterativnog poboljšanja tokom koje se čvorovi iz $C_0(X)$ dodaju u trenutnu kliku X, i faze plato pretrage tokom koje se čvorovi iz $C_1(X)$ zamenjuju sa čvorovima iz X sa kojima ne dele ivicu. Faza pretrage se završava kada su ili $C_0(X) = \emptyset$ ili $C_1(X) = \emptyset$, ili kada su svi čvorovi koji su u $C_1(X)$ već bili elementi od X tokom trenutne iteracije. Za razliku od klasičnog VNS i GVNS algoritma, implementirana je perturbacija, kao funkcija za razmrdavanje koja će biti opisana kasnije, kao poslednji korak kako bi se generisala nova početna tačka za pretragu. Iteracije se izvode sve dotle dok se ili ne pronadje MC (MVWC) ili dok se broj dozvoljenih selekcija (dodavanja u trenutnu kliku) ne dostigne.
Inicijalno, postavljeno je 50 iteracija za obavljanje VNS. Nakon toga, implementirane su različite metode selekcije čvorova za svaki podalgoritam u okviru funkcije Select; u okviru funkcije Perturb su implementirane različite perturbacije i zamene na kraju svake faze u okviru svakog pod-algoritma. Ažuriranje penala (engl. Update penalties) se izvodi tokom izvršavanja svih podalgoritama ali se penali koriste samo za selekciju čvorova dok je podalgoritam penala aktivan. Tokom razmrdavanja menja se okolina za pretragu kako bi se dostigao globalni maksimum. Razvijena je funkcija za razmrdavanje koja slučajno isključuje k čvorova iz trenutne klike. Broj čvorova koji su isključeni progresivno raste dok svi čvorovi ne budu isključeni iz klike i u tom trenutku algoritam počinje novu pretragu. Algoritam prestaje kad se definisano vreme dostiglo ili kad su se iteracije algoritma izvršile odredjen broj puta. Rešenje ove implementacije se predstavlja kao skup (niz) indeksa čvorova.

Podalgoritam penala

Svrha penala je da se dostigne dodatna diversifikacija u procesu pretrage, zbog toga što proces pretrage može da stagnira u slučaju kada graf ima više maksimalnih klika (funkcija cilja može da ima više udaljenih lokalnih optimuma). Verovatno, najočigledniji pristup za izbegavanje ove vrste stagnacije u pretrazi je da se restartuje proces konstruktivne pretrage iz drugog inicijalnog čvora. Kako bilo, čak iako postoji slučajna ili sistematična varijacija izbora inicijalnog čvora i dalje postoji rizik da je heurističko usmeravanje izgradjeno na mehanizmu pohlepne konstrukcije koja utiče na pristrasnost odredjivanja ograničenog skupa podoptimalnih klika. Celobrojni penali su povezani sa čvorovima koji moduliraju heurističku funkciju selekcije korišćenu u konstrukciji pohlepne procedure. Ovom procedurom čvorovi koji se često ubacuju u trenutnu aproksimaciju maksimalne klike dobijaju penal koji ce sprečiti heuristike koje selektuju čvorove za ubacivanje u kliku da ih ponovo odaberu. U okviru funkcije Select bira se čvor koji ima najveću vrednost funkcije prioriteta koja je definisana tako da ima velike vrednosti za čvorove koji imaju visok stepen, a dodatno penalizuje čvorove koji su se često pojavljivali u aproksimaciji maksimalne klike. U slučaju da više čvorova ima najveću vrednost funkcije prioriteta, na slučajan način bira se jedan od njih.
Uslovi zaustavljanja su maksimum CPU vremena, maksimum broja iteracija i maksimum broja iteracija izmedju dva poboljšanja.

Algoritam 12 Aproksimacije maksimalne težinske klike

Generiši inicijalno rešenje X pohlepnom heuristikom;
$X = VND(X)$; poboljšaj pohlepno rešenje;
$X^* = X$; trenutno rešenje postaje najbolje rešenje;
$k = 1$;
repeat
  Razmrdavanje (generiši tačku X na slučajan način iz $k$te okoline od X^*)
  $X := Lokalnapretraga(X)$
  $X := Platopretraga(X)$
  if $W(X) > W(X^*)$ **then**
    $X^* = X$
    $k = 1$ (resetuj brojač okoline)
  else
    $k = k + 1$(povećaj brojač okolina)
  end if
  if $k = k_{max}$ **then**
    $k = 1$
  end if
until Uslov zaustavljanja je dostignut
return X^*

Algoritam 13 Lokalna pretraga

if $X = \emptyset$ **then**
  Slučajno selektuj čvor i postavi ga u X
end if
while $C_0(X) \neq \emptyset$ **do**
  $v = Select(C_0(X))$
  $X = X \cup \{v\}$
end while
return X

Algoritam 14 Plato pretraga

loop
  $X := Lokalnapretraga(X)$
  if $C_1(X) = \emptyset$ **then return** X
  end if
  $v := Select(C_1(X))$
  if failed **then**
    $u = Nepovezancvor(X, v)$
    $X = X \cup \{v\} - u$
  end if
end loop

5.1.1 Eksperimentalni rezultati

Ova implementacija je testirana na malim/srednjim/velikim primerima sa poznatim optimalnim rešenjima ili aproksimativnim optimalnim rešenjima. Ovi skupovi testova se sastoje od popularnih benchmark primera koji se često koriste za ocenu algoritama klike. Podsećanja radi, skup benchmark DIMACS primera koji se koriste je postavljen na Drugom DIMACS Izazovu Implementacija (Second DIMACS Implementation Challenge). Ovaj skup sadrži 80 instanci grafova dobijenih iz realnih aplikacija, slučajno generisanih grafova i grafova čija je maksimalna klika sakrivena uključivanjem čvorova malog stepena. Instance ovih problema su u rangu veličine od 50 čvorova i 1000 ivica do 3300 čvorova i 5000000 ivica.
Skup od 40 BHOSLIB (engl. Benchmark with hidden optimum solutions) instanci su nastale iz SAT04 Competition dogadjaja. BHOSLIB instance su translirane iz teških slučajnih SAT problema a poznato je da su oni teški i teorijski i praktično. BHOSLIB-W i DIMACS-W težinski benchmark primeri se sastoje od primera koji se koriste za testiranje algoritama za MVWC problem. Težinski DIMACS-W primeri su dobijeni na osnovu DIMACS primera instanci alociranjem težina na čvorovima. Postoje različiti načini da se definiše funkcija težine. Primenjena je metoda opisanu u radu [197]: za svaki čvor i, težina je data sa $w_i = i \mod 200$. Generisano je 80 DIMACS instanci koje su korišćene u problemima teorije kodiranja, problemima dijagnoze grešaka, Kelerove konjekture korišćenjem hiperkocke, itd. Slično, primenjuje se ista težinska funkcija na netežinske BHOSLIB benchmark instance kako bi se dobile težinske instance (označene sa BHOSLIB-W benchmarks). Test instance su predložene od strane različith autora koji su koristili različite procedure za njihovo kreiranje. Programirano je u jeziku C++, na Intel Core 2 Duo CPU E6750 na 2.66GHz sa RAM=8Gb pod Linux Slackware 12 Kernel: 2.6.21.5, gcc version 4.1.2. Poredjena su vremena izvršavanja programa i kvalitet rešenja tj. težina nadjene klike. Eksperimentalni rezultati su predstavljeni u Tabeli 5.1. Prva kolona u tabeli sadrži test instance, druga kolona i treća kolona predstavljaju pronadjenu veličinu klike MC problema za VNS algoritam kao i CPU vreme koje je trebalo programu da pronadje rešenje, respektivno. Kolona četiri predstavlja veličinu maksimalne klike koju je postigao algoritam Fazne lokalne pretrage (engl. Phased local search, PLS) [197]. Kolonom pet predstavljeno je odgovarajuće CPU vreme koje PLS algoritam koristi. Poslednje dve kolone predstavljaju veličinu maksimalne klike koje je postigao algoritam Tabu pretrage više okolina (engl. Multi-neighborhood tabu search for the maximum weight clique problem, MNTS) [223] i odgovarajuće CPU vreme. Maksimalan dozvoljen broj iteracija je označen parametrima Itermax per run i Itermax per instance, i postavljene su na 100 za PLS i MNTS, a na 50 za VNS. Kako se može videti iz tabele, VNS je u mogućnosti da dostigne optimalno rešenje za sve test instance za značajno manje CPU vreme. U nekoliko primera, dobijen je bolji pretpostavljeni maksimum nego PLS algoritam. Kriterijum zaustavljanja vezan za CPU vreme je postavljen na 1 sekundu. Iako je PLS algoritam u mogućnosti da dostigne optimalno rešenje za najviše instanci potrebno mu je znatno više vremena za izvršavanje nego VNS algoritmu. Testiran je VNS algoritam na DIMACS i BHOSLIB netežinskim primerima s obzirom da su PLS i MNTS algoritmi takodje testirani na istim primerima. Svaka instanca je rešavana 100 puta nezavisno putem MNTS algoritma sa različitim slučajnim seed grupama. Dubina pretrage u MNTS algoritmu, označena je sa L i dodeljena joj je vrednost 4000 za instance težinskog slučaja MVWC. Za netežinski slučaj MCP, parametar L ima vrednost 104, osim za brock i san familije primera (DIMACS) za koje L ima vrednost 100. Glavni kriterijum uporedjivanja je bio kvalitet pronadjenog rešenja. CPU vremena su poredjena u odnosu na razlike u programskim jezicima, strukture podataka, kompajlere, procesore, itd. Tabela se sastoji od 6 kolona sa rezultatima i jednom kolonom sa imenom instance. Prvo su predstavljeni VNS rezultati, zatim PLS rezultati, a konačno MNTS rezultati. Za svaki algoritam data je maksimalna veličina klike i CPU vreme da se ista pronadje. Vrednost u zagradi jeste u koliko restartovanja programa je dostignuta optimalna veličina. U Tabeli 5.1 su istaknuti primeri u kojima su rezultati bolji u odnosu na rad [197]. Poslednji red u tabeli jeste prosečna vrednost klike koja je pronadjena u odnosu na sve primere, kao i prosečno vreme, uz prosečne vrednosti ostala dva algoritma sa kojima su poredjeni.

Test	**VNS**	**VNS**	**PLS**	**PLS**	**MNTS**	**MNTS**
Instance	**VNS opt. vel.**	**VNS CPU vreme**	**PLS opt. vel.**	**PLS CPU vreme**	**MNTS opt. vel.**	**MNTS CPU vreme**
*brock*200_1	2821(50)	5.30	2821(100)	0.19	2821(100)	< 0.0
*C*1000.9	9135(2)	81.32	8965(5)	344.74	9254(100)	8.9
*C*125.9	2529(50)	5.02	2529(100)	8.08	2529(100)	0.02
*C*2000.5	2466(6)	53.19	2466(18)	711.27	2466(100)	1.84
*C*2000.9	10712(1)	350.00	10028()		10999(22)	168.11
*C*250.9	5092(32)	9.91	5092(17)	247.69	5092(100)	0.06
*C*4000.5	2770(10)	129.63	2792(−)		2792(100)	80.56
*C*500.9	**6955(10)**	26.69	6822(−)		6955(100)	0.07
*cfat*2001	1284(50)	0.60	1284(100)	$< \epsilon$	1284(100)	0.14
*cfat*2002	2411(50)	0.78	2411(100)	$< \epsilon$	2411(100)	0.06
*cfat*2005	5887(50)	1.43	5887(100)	$< \epsilon$	5887(100)	0.02
*cfat*50010	11586(50)	3.93	11586(100)	$< \epsilon$	11586(100)	0.06
*cfat*5001	1354(50)	1.20	1354(100)	$< \epsilon$	1354(100)	0.73
*cfat*5002	2628(50)	1.37	2628(100)	0.01	2628(100)	0.33
*cfat*5005	5841(50)	2.37	5841(100)	$< \epsilon$	5841(100)	0.14
*DSJC*1000.5	2186(16)	14.78	2186(100)	47.76	2186(100)	0.20
*DSJC*500.5	1725(7)	6.47	1725(100)	0.95	1725(100)	0.04
*gen*200_*p*0.9_44	5043(50)	8.23	5043(100)	4.44	5043(100)	< 0.01
*gen*200_*p*0.9_55	5416(50)	6.08	5416(100)	0.05	5416(100)	0.33
*gen*400_*p*0.9_55	6718(9)	22.94	6718(2)	340.11	6718(100)	0.15
*gen*400_*p*0.9_65	6940(33)	20.90	6935(4)	200.79	6940(100)	0.04
*gen*400_*p*0.9_75	8006(50)	14.55	8006(100)	$< \epsilon$	8006(100)	0.88
*hamming*102	50512(50)	99.47	50512(100)	$< \epsilon$	50512(100)	0.92
*hamming*104	**5127(1)**	45.68	5086(1)	1433.07	5129(100)	2.21
*hamming*62	1072(50)	1.24	1072(100)	$< \epsilon$	1072(100)	< 0.01
*hamming*64	134(50)	0.36	134(100)	$< \epsilon$	134(100)	< 0.01
*hamming*82	10976(50)	6.99	10976(100)	$< \epsilon$	10976(100)	< 0.01
*hamming*84	1472(50)	3.17	1472(100)	$< \epsilon$	1472(100)	< 0.01
*johnson*1624	548(50)	1.26	548(100)	$< \epsilon$	548(100)	0.23
*johnson*3224	2033(50)	7.13	2033(100)	44.68	2033(100)	0.53
*johnson*824	66(50)	0.24	66(100)	$< \epsilon$	66(100)	< 0.01
*johnson*844	511(50)	1.03	511(100)	$< \epsilon$	511(100)	< 0.01
*keller*4	1153(48)	2.37	1153(100)	0.02	1153(100)	0.03
*keller*5	3317(9)	19.45	3317(100)	119.24	3317(100)	3.17
*keller*6	**8062(1)**	350.00	7382()		8062(5)	606.15
*MANN_a*27	**12282(1)**	127.66	12264()		12281(1)	88.28
*MANN_a*81	**111196(1)**	351.58	110564()		111128(1)	832.24
*MANN_a*9	372(50)	1.45	372(100)	$< \epsilon$	372(100)	< 0.01
*p_hat*10001	1514(5)	8.83	1514(100)	7.61	1514(100)	0.08
*p_hat*10002	5777(50)	31.65	5777(87)	940.62	5777(100)	0.11
*p_hat*10003	8111(38)	49.14	7986()	940.62	8111(100)	1.23

p_hat15001	1619(17)	14.03	1619(100)	48.91	1619(100)	0.06
p_hat15002	7360(42)	70.17	7328(4)	1056.19	7360(100)	0.82
p_hat15003	10321(37)	233.54	10014()		10321(96)	188.38
p_hat3001	1057(43)	2.88	1057(100)	0.01	1057(100)	0.02
p_hat3002	2487(50)	8.07	2487(100)	19.36	2487(100)	< 0.01
p_hat3003	3774(49)	11.50	3774(47)	418.11	3774(100)	0.02
p_hat5001	1231(6)	4.91	1231(100)	0.42	1231(100)	0.03
p_hat5002	3920(50)	15.66	3925()		3920(100)	< 0.01
p_hat5003	5375(42)	24.97	5361()		5375(100)	0.10
p_hat7001	1441(50)	7.44	1441(100)	0.20	1441(100)	0.03
p_hat7002	5290(50)	23.11	5290(100)	78.51	5290(100)	0.02
p_hat7003	7565(50)	33.73	7565(12)	718.40	7565(100)	0.38
san1000	1716(6)	32.11	1716()		1716(100)	13.01
san200_0.7_1	3370(40)	6.86	3370(100)	$< \epsilon$	3370(100)	0.17
san200_0.7_2	2422(43)	8.78	2422(66)	397.38	2422(100)	0.02
san200_0.9_1	6825(50)	7.52	6825(100)	$< \epsilon$	6825(100)	0.13
san200_0.9_2	6082(50)	4.48	6082(100)	$< \epsilon$	6082(100)	0.21
san200_0.9_3	4748(48)	9.81	4748(72)	219.68	4748(100)	< 0.01
san400_0.5_1	1455(16)	14.23	1455(100)	200.44	1455(100)	0.06
san400_0.7_1	3941(50)	26.85	3941(100)	0.03	3941(100)	13.68
san400_0.7_2	2952(46)	18.57	3110(100)	0.05	3110(100)	43.34
san400_0.7_3	2771(14)	14.65	2771(100)	4.41	2771(100)	0.05
san400_0.9_1	9776(9)	53.29	9776(100)	$< \epsilon$	9776(100)	1.29
sanr200_0.7	2325(50)	4.35	2325(100)	0.62	2325(100)	< 0.01
sanr200_0.9	5126(50)	8.50	5126(5)	182.54	5126(100)	< 0.01
sanr400_0.5	1835(13)	6.09	1835(100)	0.67	1835(100)	0.02
sanr400_0.7	2992(11)	9.78	2992(100)	141.50	2992(100)	< 0.01
prosek	6522.29(35)	37.51	6483.23(83)	153.09	6529.92(94)	30.28

Tabela 5.1: Poredjenje rezultata eksperimenata MVWCP

Algoritam VNS primenjen na težinski slučaj je dostigao znatno bolje rezultate i znatno kraće vreme nego što je to do tada bilo prikazano u state-of-the-art PLS algoritmu. Osim toga, VNS je bio relativno jednostavan za primenu na problem MVWC, i kao takav bio je odličan za početak istraživanja klike.
MNTS je state-of-the-art algoritam za pomenuti problem, zahvaljujući dobro odabranim okolinama i redosledu okolina u primeni. Očekivano je da dobrim odabirom okolina VNS dostigne još bolje rezultate i time nadmaši trenutni state-of-the-art. Testiranje algoritma se može usmeriti na VND, odnosno može se ukinuti faza razmrdavanja čime bi rešenje zavisilo isključivo od rasporeda okolina, sve dok ima poboljšanja. Takođe, testiranje se može usmeriti na RVNS, odnosno ukidanjem faze lokalne pretrage moguće je ispitati po jedno slučajno rešenje u svakoj od okolina. Oba smera istraživanja dovode do analize ponašanja okolina u odnosu na ubrzavanje rešenja.
Mora se naglasiti da je VNS u tri primera bolji od ovog trenutno najboljeg algoritma, a to su: *keller*6, $MANN\_a27$ i $MANN\_a81$. U $55,38\%$ slučajeva VNS je dostigao maksimalan broj puta identične veličine klike (maksimume). U $92,31\%$ slučajeva VNS je dostigao maksiume klike bez obzira na broj ponavljanja. Što se vremena tiče, MNTS je i dalje bolji u velikoj većini, iako se to može objasniti brzinom procesora i zauzećem memorije. Ipak, zaključak je da postoji veliki potencijal u daljem razvoju VNS algoritma za problem MVWC s obzirom da je u velikoj većini primera VNS dostigao solidne rezultate za težinski slučaj maksimalne klike. Pravac poboljšanja će ići ka raspodeli okolina za kandidate klike pri čemu će se selektovati oni čvorovi sa strožijom funkcijom prioriteta h i sa strožijim kriterijumom za vreme provedeno u kliki.

5.2 Problem particije klike

Problem particije klike (CPP) se može reformulisati kao problem maksimalnog različitog grupisanja (MDGP). Testirano je nekoliko verzija VNS algoritma i odlučeno da se upotrebi SVNS algoritam, kasnije modifikovan u Adaptivnu generalnu metodu promenljivih okolina (engl. Skewed general variable neighborhood search, SGVNS) čime su dobijena značajna poboljšanja u odnosu na "state of the art" metode koje su rešavale velike instance. Ovo je dalje potvrdilo korist diversifikacije koju daje kombinovani pristup SVNS i lokalne pretrage iz GVNS [32].

SGVNS heuristika koja je predstavljena u ovom radu koristi se za rešavanje CPP problema veoma efikasno. Kombinuje se diversifikacija koja se dobija preko Adaptivne VNS metode sa intenzifikacijom VND, a lokalna pretraga se obavlja kao u GVNS. Kombinovani pristup se skorije uspešno koristio sa povezanim MDGP [32]. Dostignut je sličan uspeh kod CPP sa značajnim poboljšanjem kvaliteta rešenja i predstavljen je rezultat na velikim primerima uporedjujući ih sa "state of the art" rešenjima.

5.2.1 Opis problema

Neka je dat težinski, neorijentisan i kompletan graf $G = (V, E, c)$, potrebno je odrediti particiju skupa čvorova na proizvoljan broj, disjunktnih podskupova $V_1, V_2, ..., V_k$ tako da je suma težina ivica u okviru podgrafova (klika) $G_1, G_2, ..., G_k$, indukovanih podskupovima $V_1, V_2, ..., V_k$ minimalna. Broj podskupova nije fiksiran, tako da je potrebno odrediti onu podelu čvorova grafa pri kojoj je suma ivica minimalna. Neka je c_{ij} težina (trošak) ivice (i, j) za sve parove čvorova (i, j), $1 \leq i < j \leq n$, gde je n jednako broju čvorova u $G(n = |V|)$. Matematička formulacija problema je sledeća [39]:

$$
\begin{aligned}
&\min \sum_{(i,j)\in E} c_{ij}x_{ij} \\
&\text{s.t.} \\
&x_{ij} + x_{jr} - x_{ir} \leq 1, \qquad \forall 1 \leqslant i < j < r \leqslant n \\
&x_{ij} - x_{jr} + x_{ir} \leq 1, \qquad \forall 1 \leqslant i < j < r \leqslant n \\
&-x_{ij} + x_{jr} + x_{ir} \leq 1, \quad \forall 1 \leqslant i < j < r \leqslant n \\
&x_{ij} \in \{0, 1\}, \qquad \forall 1 \leqslant i < j \leqslant n.
\end{aligned} \tag{5.1}
$$

Ako je $x_{ij} = 1$, onda su čvorovi i i j u istom klasteru, i težina c_{ij} ivice (i, j) se dodaje u funkciju cilja; inače se ne dodaje. Skup ograničenja osigurava da su sve ivice u kompletnom

podgrafu $G_t, t = 1, 2, ..., k$ uključene u rešenja. Problem particije klike može biti proširen na nekompletan graf uključivanjem veštačkih ivica sa velikom pozitivnom težinom izmedju parova čvorova gdegod nedostaju ivice u originalnom grafu. Velike pozitivne težine osiguravaju da svaki podgraf $G_t, t = 1, 2, ..., k$ bude klika koja ne sadrži izmišljene ivice. Ako sve ivice u kompletnom grafu G imaju negativne (nula) težine, problem postaje trivijalan, a optimalno rešenje se dobija za $k = 1$ i $G_1 = G$. Tako, problem CPP postaje interesantan samo kad neke od težina ivica imaju pozitivne vrednosti. U CPP se ne zna broj grupa, i ne postoji limit u veličini grupa.
CPP se često javlja u društvenim naukama [39]. Wang i drugi [220] su pokazali da se CPP povoljno rešava preko K-means algoritma, koji analizira prividne klase i klaster strukture u realnim bazama podataka. Jedna od koristi ovog modela je da se ne mora znati broj klastera unapred. Poznata citirana primena modela se ogleda u agregaciji relacija binarne ekvivalencije. Parametri c_{ij} reprezentuju broj atributa u odnosu na koje se i i j ne slažu umanjen za broj atributa u odnosu na koje se slažu. Ukoliko na primer postoji 10 atributa koji se svi uporedjuju i neka su dati i i j koji imaju identičnu meru na 8 od njih, parametar $c_{ij} = 2 - 8 = -6$ označava da će se ta dva čvora naći u istom klasteru. U slučaju da je broj koji se dobije pozitivan, tendencija je da se čvorovi nadju u različitim klasterima.
Najranija metoda za rešavanje CPP koristi jednostavnu proceduru relociranja procedura promene čvorova preko klastera dok lokalni optimum nije postignut [199]. Lokalna pretraga relocira okoline oko jednog čvora. De Amorim i drugi [72] su predložili simulirano kaljenje i tabu pretragu, a bazirali su ih na relokaciji okolina. To je uključilo mogućnost relokacije čvora u prazan skup i tako se broj klastera povećao za jedan. Charon i Hudry [54] su istraživali različite procedure šuma kojima su deformisali podatke sa namerom da stvore uzlazne putanje. Brusco i Kohn [39] su razvili "state-of-the-art" Heuristiku pretraga okolina, prateći ideju VNS metaheuristike [109], gde su razmrdavanje (perturbaciju) realizovali na slučajan način umesto na sistematičan, kao u VNS.
Pokazano je kako CPP može biti formulisan kao problem maksimalnog različitog grupisanja. To omogućava modifikaciju Adaptivnog generalnog VNS koji je skorije razvijen i njegovu primenu na CPP. Brusco i Kohn [39] su publikovali slične rezultate sa svojom Heuristikom pretraga okolina na test instancama iz rada Charon i Hudry [54], mada su im značajno bolji rezultati nego što su publikovali Charon i Hudry na većim instancama problema. Dostignut je sličan rezultat u novijim radovima.

Veza sa problemom maksimalnog različitog grupisanja

MDGP se bavi pretragom particija datog skupa elemenata u specifičan broj uzajamno disjunktnih podskupova kako bi se maksimizovao ukupni diverzitet izmedju elemenata iste grupe. Jedna od najranijih aplikacija MDGP bila je formiranje studentskih radnih grupa. Važno je skup podeliti na različite studijske grupe kako bi se poboljšali uslovi učenja. Druga aplikacija formirala je jednake grupe recenzenata kako bi se evaluirali istraživački predlozi [123]. Cilj je bio formirati različite grupe kako bi se osiguralo da su projekti evaluirani sa nekoliko različitih pogleda na problematiku. Ostale aplikacije se mogu videti u literaturi [158].
U MDGP broj grupa je fiksiran, i tipično ove grupe moraju imati istu ili blisku veličinu. Kako bi se rešila ova situacija, postavlja se broj grupa inicijalno na gornju granicu od n, i dozvoljava se da neke grupe budu prazne. Kako nije bilo nametnutih ograničenja u veličini grupa, broj grupa je ostavljen da se indirektno odredi u formulaciji ovog modela i on predstavlja broj nepraznih grupa u optimalnom rešenju. Neka je $y_{ig} = 1$, ako čvor i pripada grupi g i 0, inače. Reformulisan je CPP preko sledećeg problema kvadratnog binarnog programiranja:

$$\begin{aligned} &\max -\sum_{g=1}^{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}y_{ig}y_{jg} \\ &\text{s.t.} \\ &\sum_{g=1}^{n} y_{ig} = 1, && i = 1, ..., n \\ &y_{ig} \in \{0, 1\}, && i = 1, ..., n, g = 1, ..., n \end{aligned} \tag{5.2}$$

Ova formulacija je ekvivalentna generalnom MDGP gde su izbačena sva ograničenja na veličinu grupa i broj grupa nije naznačen. Ovde je broj grupa samo implicitno naznačen da je n ali je dozvoljeno da neke grupe budu prazne. Nelinearna formulacija CPP 5.2 dozvoljava

Redukciju u broju ograničenja, za razliku od originalne linearne Formulacije 5.1. Broj binarnih varijabli ostaje isti u obe Formulacije 5.1 i 5.2. Preformulisanjem modela u MDGP, bilo je moguće preuzeti bilo koju metodologiju rešavanja za MDGP i primeniti je na rešavanje CPP. Adaptirana je nova Adaptivna GVNS heuristika koja je primenjena sa velikim uspehom na MDGP [32].

Prostor rešenja CPP

Prostor rešenja se sastoji od svih mogućih podela elemenata u grupe. Rešenje je predstavljeno preko vektora x^c dužine $n, (x^c = (x_1{}^c, x_2{}^c, ..., x_n{}^c))$, tako da je $x_i{}^c$ oznaka grupe koja sadrži element $i, (i = 1, 2, ..., n)$. Kako bi se ubrzala lokalna pretraga, kreira se matrica sc^c, tako da je sc^c_{ig} suma težina ivica izmedju elementa i i svih elemenata koji su pridruženi grupi g u trenutnom rešenju:

$$sc^c_{ig} = \sum_{j=1,2,...,n; x^c_j = g} c_{ij}.$$

Može se primetiti da je za trenutno rešenje, matrica sc^c izračunata za $O(n^2)$ vreme. Inicijalno, broj grupa je n. Ukoliko su u trenutno najboljem rešenju elementi podeljeni na g' grupa, pretpostavlja se da postoji jedna prazna grupa ($g_{\max} = g' + 1$).

VND lokalna pretraga

Lokalna pretraga je implementirana kao Metoda spusta kroz promenljive okoline (VND), i dizajnirane su sledeće okoline u okviru njega: uključivanje i zamena (engl. Insertion, Swap). Okolina dobijena uključivanjem sadrži sva rešenja koja se dobijaju pomeranjem samo jednog elementa iz trenutne grupe u drugu grupu. Koristeći prethodno definisanu matricu sc^c, moguće je da se efikasno izračuna za svaki dopustivi pokret, promena vrednosti funkcije cilja. Neka je sa x^n označeno rešenje koje se dobija pošavši od rešenja x^c, pomeranjem elementa i iz trenutne grupe g_1 u grupu g_2. Suma težina ivica u svim grupama osim u grupama g_1 i g_2 je nepromenjena. Element i je pomeren iz grupe g_1 i zbog toga suma težina ivica u grupi g_1 je opala za sumu težina ivica izmedju i i svih drugih elemenata koji postoje u g_1. Element i je uključen u g_2, tako da je suma težina ivica u g_2 porasla za sumu težina svih ivica izmedju i i elemenata koji pripadaju grupi g_2. Jednostavno je zaključiti da je razlika izmedju vrednosti funkcije cilja za rešenja x^c i x^n:

$$\Delta f = f(x^n) - f(x^c) = sc^c_{ig_2} - sc^c_{ig_1}.$$

Algoritam 15 Lokalna pretraga u okolini uključivanja

```
function LSIns(x, sc, f)
    rez ← false
    for v ← 1 to n do
        for g ← 1 to g_max do
            if x_v ≠ g then
                df ← sc_{v,g} − sc_{v,x_v}
                if df < 0 then
                    Updatesc (x, sc, v, g)   x_v ← g   f ← f + df
                    rez ← true
                end if
            end if
        end for
    end for
return rez
end function
```

Kada je izvodjen korak uključivanja, menjano je trenutno rešenje, i tada je bilo neophodno da se osveži matrica sc^c. Kada se element i pomerao iz grupe g_1 u grupu g_2, tada su grupe g_1 i g_2 modifikovane i vrednosti $sc^c_{jg_1}$ i $sc^c_{jg_2}$ su bile osvežavane na sledeći način:

$$sc^c_{jg_1} = sc^c_{jg_1} - c_{ji}$$

i

$$sc^c_{jg_2} = sc^c_{jg_2} + c_{ji}.$$

Kako je osvežavanje izvodjeno za svaki element j, osvežavanje matrice sc^c ima složenost $O(n)$ nakon izvodjenja pokreta uključivanja. Sa druge strane, kardinalnost okolina za uključivanje je $O(g_{\max}n)$, gde je $g_{\max}$ trenutni broj grupa. Okolina za zamenu sadrži sva rešenja koja se dobijaju razmenom tačno jednog para elemenata koji pripadaju različitim grupama. Neka je element i u grupi g_i i element j u grupi g_j u trenutnom rešenju x^c. Neka je x^n rešenje koje se dobija nakon pomeranja elementa i u grupu g_j i elementa j u grupu g_i. Kako je element i pomeren iz grupe g_i , težina ivica izmedju elementa i i elemenata koji preostaju u grupi g_i ne utiče na funkciju cilja novog rešenja. Kako je element i uključen u grupu g_j, težina ivica izmedju i i elemenata koji pripadaju grupi g_j utiče na vrednost funkcije cilja novog rešenja. Slična činjenica važi za element j. Konačno, može se izračunati razlika izmedju vrednosti funkcije cilja trenutnog i okolnih rešenja:

$$\Delta f = f(x^n) - f(x^c) = (sc^c_{ig_j} - sc^c_{ig_i}) + (sc^c_{jg_i} - sc^c_{jg_j}) - 2c_{ij}.$$

Očigledno je da se promena u vrednosti funkcije cilja za svako rešenje iz okoline zamene obavlja za $O(1)$ vreme, dok je kardinalnost zamene izračunata za $O(n^2)$. Nakon izvodjenja pokreta zamene neophodno je osvežiti matricu sc^c, a kompleksnost ovog osvežavanja iznosi $O(n)$ (zamena je kao dva sukcesivna uključivanja).

Algoritam 16 Lokalna pretraga u okolini zamene

```
function LSSwap(x, sc, f)
    rez ← false
    for v ← 1 to n do
        for u ← v + 1 to n do
            if x_v ≠ x_u then
                df ← sc_{v,x_u} + sc_{u,x_v} − sc_{v,x_v} − sc_{u,x_u} − 2d_{v,u}
                if df < 0 then
                    Swap(x, sc, v, u)
                    f ← f + df
                    Updatesc(x, sc, v, u)
                    rez ← true
                end if
            end if
        end for
    end for
return rez
end function
```

5.2.2 Adaptivni generalni algoritam promenljivih okolina

U prostoru rešenja uvodi se distanca $d(x^n, x^b)$ izmedju rešenja x^n i x^b na sledeći način:

$$d(x^n, x^b) = \frac{\left|\{(i,j)|1 \le i < j \le n, ((x^b_i = x^b_j) \wedge (x^c_i \neq x^c_j)) \vee ((x^b_i \neq x^b_j) \wedge (x^c_i = x^c_j))\}\right|}{\sum_{g=1}^{g_b} \binom{c^b_g}{2}}.$$

gde je g_b broj grupa u rešenju x^b i ${c_g}^b$ je broj elemenata u grupi g u rešenju x^b. Intuitivno, izraz u brojiocu je broj parova elemenata koji pripadaju istom klasteru (grupi) u jednom rešenju, ali ne pripadaju istoj grupi u drugom rešenju. Imenilac je jednak broju ivica koje učestvuju u kliki u optimalnom rešenju. Izraz $d(x^c, x^n)$ znači distancu izmedju rešenja x^c i x^n i izračunava se na sličan način. Pseudo kod za razmrdavanje je dat algoritmom Razmrdavanja.

5.2.3 Računarski rezultati i test primeri

Rezultati koji su predstavljeni, dobijeni su na Intel procesoru, 3.2 GHZ CPU i 4Gb RAM. SGVNS je programiran u C++ jeziku.

13 benhcmark primera

Algoritam 17 SGVNS$(\alpha, k_{min}, k_{\max}, k_{step})$

$f^c \leftarrow$ `Inicijalnoreenje`(x^c, sc^c)
`VND`(x^c, sc^c, f^c)
$(x^b, sc^b, f^b) \leftarrow (x^c, sc^c, f^c)$
$k \leftarrow k_{min}$
while Uslov zaustavljanja nije ispunjen **do**
  $(x^n, sc^n, f^n) \leftarrow (x^c, sc^c, f^c)$
  `Razmrdavanje`(k, x^n, sc^n, f^n)
  `VND`(x^n, sc^n, f^n)
  if $(f(x^n)/f(x^c) + d(x^n, x^c) > 1 + \alpha)$ `and`$(f(x^n)/f(x^b) + d(x^n, x^b) > 1 + \alpha)$ **then**
    $(x^c, sc^c, f^c) \leftarrow (x^n, sc^n, f^n)$
    if $f(x^c) < f(x^b)$ **then**
      $(x^b, sc^b, f^b) \leftarrow (x^c, sc^c, f^c)$;
    end if
    $k \leftarrow k_{min}$
  else
    $k \leftarrow k + k_{step}$
    if $k > k_{\max}$ **then**
      $k \leftarrow k_{min}$
    end if
  end if
end while

Algoritam 18 Razmrdavanje

procedure RAZMRDAVANJE(k, x, sc, f)
  Brisati $\min\{k, 30\}$ čvorove i pomeriti u prazni skup
  while $k > 0$ **do**
    $(u, v) \leftarrow$(`RandVert, RandVert`)
    if $x_u \neq x_v$ **then**
      `Swap`(x, u, v)
      $k \leftarrow k - 1$
    end if
  end while
  `Computesc`(x, sc)
  $f \leftarrow$ `Computeobj`(x)
end procedure

Prvi skup se sastoji od 7 benchmark primera koje su dali [54] (rand100-100, rand300-100, rand500-100, rand300-5, zahn300, sym300-50, regnier300-50) i 6 primera koje je generisao Brusco (rand200-100, rand400-100, rand100-5, rand200- 5, rand400-5, rand500-5). Za više detalja videti [39].

30 velikih instanci

Kako bi se uporedjivale heuristike na velikim problemima predložene su nove CPP instance. Sastoje se od skupova od 1000, 1500 i 2000 čvorova, a svaki ima 10 instanci. Dužina ivice je generisana slučajno kao uniformna distribucija celih brojeva u rangu T $[-100, 100]$.

Preliminarno testiranje i vrednosti parametara

Kao što je objašnjeno ranije u pseudokodu, SGVNS ima 4 parametara: α, k_{min}, $k_{\max}$, k_{step}. Vrednosti tri parametra za definiciju okolina su: $k_{\max} = \max\{100, n/5\}$, $k_{min} = k_{step} = \max\{1, k_{\max}/50\}$.

Vrednost α (koristi se u fazi uvrtanja SGVNS) odredjena je eksperimentalno tako što je heuristika izvršavana po 10 puta na instancama sa $n = 1000$ i sa različitim vrednostima α, npr., $\alpha \in \{0.05, 0.10, 0.20, 0.30, 0.40, 0.50\}$. Rezultati su u Tabeli 5.2.

Iz tabele se zaključuje da je najbolja vrednost parametra $\alpha = 0.40$, i koristi se u preostalim testovima.

Uporedjivanje sa "state-of-the-art" heuristikama

Tabela 5.2: SGVNS rezultati na `rand1000_01` primerima na deset restarta sa različitim vrednostima α

	SGVNS		
α	f_{best}	f_{avg}	Time
0.05	-864563	-853351.10	688.70
0.1	-868994	-860151.00	562.56
0.2	-874312	-869298.10	649.18
0.3	-874975	-871983.00	761.78
0.4	-876077	-873107.60	737.99
0.5	-874416	-873123.50	719.01

Porede se rezultati koji su dobijeni preko SGVNS sa ostalim heuristikama iz literature. "State-of-the-art" su NS-R (engl. Neighborhood search-Relocation) i NS-TS (engl. Neighborhood search-Tabu search), obe su predložili Brusco i Köhn [39]. U Tabeli 5.3 porede se 3 heuristike na 13 malih test primera. Druga kolona pokazuje najbolje poznate vrednosti koje su dobijene preko `SGVNS` u 600 izvršavanja. Vrednosti funkcije cilja dobijene iz `NS-R` i `NS-TS` date su u kolonama 3 i 4 respektivno. Ostale 3 kolone predstavljaju rezultate koje su dobijeni preko `SGVNS`: najbolja vrednost funkcije cilja u 10 izvršavanja, prosečna vrednost i prosečno CPU vreme potrebno za nalaženje najboljeg rešenja. Vreme izvršavanja druge dve heuristike je postavljeno na 500 sekundi.

Tabela 5.3: Uporedjivanje `NS-R`, `NS-TS` i `SGVNS` na malim primerima; `SGVNS` deset restarta i $t_{\max} = n$.

				SGVNS		
name	Best	`NS-R`	`NS-TS`	f_{best}	f_{avg}	Time
rand100-5	-1407	-1407	-1407	-1407	-1407	0.33
rand100-100	-24296	-24296	-24296	-24296	-24296	1.42
rand200-5	-4079	-4079	-4079	-4079	-4079	26.59
rand200-100	-74924	-74924	-74924	-74924	-74924	12.56
rand300-5	-7732	-7723	-7729	**-7732**	-7728	87.12
rand300-100	-152709	-152709	-152709	-152709	-152709	24.81
sym300-50	-17592	-17592	-17592	-17592	-17592	143.45
regnier300-50	-32164	-32164	-32164	-32164	-32164	3.24
zahn300	-2504	-2503	**-2504**	**-2504**	-2504	29.4
rand400-5	-12133	-12096	-12120	**-12133**	-12123	206
rand400-100	-222757	-222647	-222374	**-222757**	-222735	212.65
rand500-5	-17127	-17008	-17086	**-17127**	-17095.5	255.29
rand500-100	-309125	-308620	-308341	**-309107**	-308754	291.00
prosek	**-67580.69**	-67521.30	-67486.53	**-67579.30**	**-67546.96**	**99.52**

Rezultati za manje instance su dobijeni sa jednakim kvalitetom za sve heuristike. Za veće instance `SGVNS` je dao bolje rezultate. U Tabeli 5.4 iste metode su uporedjivane na novim velikim test instancama. U poslednje dve kolone daje se % poboljšanja kod `SGVNS` u odnosu na druge dve "state-of-the-art" heuristike.

Za sve metode vremensko ograničenje je postavljeno u odnosu na broj čvorova. U svih 30 velikih instanci, `GVNS` rešenja su boljeg kvaliteta i zahtevaju manje CPU vreme. To znači da se nova SGVNS heuristika može smatrati kao "state of the art" za CPP.

Tabela 5.4: Rezultati velikih instanci bazirani na 10 izvršavanja

	NS-R			NS-TS			SGVNS			%Im over	
Name	f_{best}	f_{avg}	$time$	f_{best}	f_{avg}	$time$	f_{best}	f_{avf}	$time$	NS-R	NS-TS
r1000_01	-867120	-864600.30	1054.31	-866105	-865510.70	1033.85	**-876077**	-873107.60	737.99	1.02	1.14
r1000_02	-867506	-866265.30	1048.33	-872131	-867497.70	1113.92	**-878301**	-875259.40	824.02	1.23	0.70
r1000_03	-869632	-866454.30	1047.74	-867505	-862686.30	1086.74	**-878337**	-873807.30	893.15	0.99	1.23
r1000_04	-863587	-860479.30	1059.32	-864306	-860894.70	1080.27	**-874224**	-870403.30	828.44	1.22	1.13
r1000_05	-871291	-869459.30	1055.17	-873648	-872259.70	1092.70	**-887802**	-877659.50	760.26	1.86	1.59
r1000_06	-883498	-880247.60	1043.91	-888777	-883757.30	1070.99	**-890693**	-882873.40	782.65	0.81	0.22
r1000_07	-874515	-873598.30	1069.01	-874095	-870816.00	1056.59	**-900474**	-883970.30	750.51	2.88	2.93
r1000_08	-877350	-873365.00	1043.38	-878923	-874766.70	1081.24	**-887370**	-876317.50	737.92	1.13	0.95
r1000_09	-884951	-883422.30	1060.23	-888638	-886901.00	1023.65	**-891663**	-887789.00	776.10	0.75	0.34
r1000_10	-868132	-865049.00	1050.94	-866735	-864425.70	1088.51	**-875397**	-872635.70	849.36	0.83	0.99
r1500_01	-1600474	-1594384.30	1596.51	-1592399	-1591123.70	1782.53	**-1613347**	-1610805.22	1185.06	0.80	1.30
r1500_02	-1583069	-1577906.30	1541.10	-1584404	-1580197.30	1649.66	**-1602612**	-1596562.22	1231.76	1.22	1.14
r1500_03	-1583597	-1581930.00	1563.01	-1591449	-1583654.00	1657.47	**-1601090**	-1595340.80	1292.90	1.09	0.60
r1500_04	-1578713	-1575948.30	1541.04	-1590351	-1580704.70	1713.55	**-1597926**	-1594072.00	1123.63	1.20	0.47
r1500_05	-1585727	-1582063.30	1553.26	-1586615	-1584059.30	1601.34	**-1603949**	-1594446.20	1240.69	1.14	1.08
r1500_06	-1582878	-1578623.30	1541.54	-1588589	-1580136.30	1670.01	**-1601118**	-1592169.60	1326.36	1.14	0.78
r1500_07	-1609452	-1601790.00	1536.98	-1617723	-1607697.00	1656.17	**-1627105**	-1615951.30	1253.96	1.08	0.58
r1500_08	-1565381	-1562638.00	1545.45	-1571224	-1563231.70	1591.34	**-1586554**	-1579111.10	1286.64	1.33	0.97
r1500_09	-1600449	-1598608.30	1557.15	-1607941	-1600318.30	1642.45	**-1625854**	-1612040.60	1251.96	1.56	1.10
r1500_10	-1593909	-1588740.30	1552.03	-1585602	1582132.30	1644.01	**1607360**	1595823.20	1042.75	0.84	1.35
r2000_01	-2467372	-2460571.60	2081.64	-2476734	-2459899.60	2237.85	**-2493143**	-2485535.50	1596.02	1.03	0.66
r2000_02	-2463804	-2451838.00	2062.19	-2467493	-2447006.10	2176.77	**-2484001**	-2473849.60	1451.65	0.81	0.66
r2000_03	-2462756	-2458149.90	2062.58	-2477969	-2460834.60	2152.34	**-2490379**	-2481812.60	1569.89	1.11	0.50
r2000_04	-2469300	-2459721.00	2051.13	-2478567	-2461141.70	2129.76	**-2497504**	-2485469.90	1633.03	1.13	0.76
r2000_05	-2445974	-2435498.50	2107.07	-2445534	-2436632.40	2365.45	**-2472230**	-2459713.90	1641.98	1.06	1.08
r2000_06	-2451637	-2436883.30	2085.50	-2449319	-2436018.60	2132.28	**-2473995**	-2462179.60	1684.78	0.90	1.00
r2000_07	-2459744	-2449809.70	2072.86	-2456281	-2448892.00	2290.89	**-2483346**	-2472279.20	1760.67	0.95	1.09
r2000_08	-2440670	-2431941.10	2065.87	-2439846	-2423618.60	1915.77	**-2467568**	-2451094.40	1818.02	1.09	1.12
r2000_09	-2468137	-2458719.80	1862.17	-2461927	-2454540.10	1924.97	**-2495666**	-2479213.90	1740.09	1.10	1.35
r2000_10	-2415438	-2409663.80	1859.33	-2422183	-2411409.50	1973.93	**-2453464**	-2437303.70	1663.52	1.55	1.27
prosek	-1638535.43	-1633278.98	1545.69	-1641100.43	-1633425.45	1621.23	**-1657284.96**	**-1648286.58**	**1224.52**	**1.16**	**1.01**

5.3 Problem netežinske klike

U poglavlju 3, MC problem formulisan je na više načina kao problem kontinualne optimizacije. Podsećanja radi kontinualna funkcija maksimalne klike je data na osnovu Motzkin-Straus [178] formule:

$$F(x) = \frac{1}{2}x^T Ax, \quad x \in S = \{x \in R^n | \sum_{i=1}^{n} x_i = 1, x_i \geq 0, i = 1, ..., n\} \tag{5.3}$$

gde je $A = A_G + \frac{1}{2}I$ i I je jedinična matrica, a A_G je matrica susedstva u grafu.
Neka je C lokalno maksimalna klika veličine K a $z = z(C)$ njen karakterističan vektor i neka je z strogi lokalni maksimum u Modelu 5.3. Tada se vrednost funkcije može izračunati preko formule

$$F(z) = \frac{1}{2} - \frac{1}{4K}$$

Kako se ovaj zadatak ne može svrstati pod oblast konveksne optimizacije s obzirom da matrica A sadrži kako pozitivne tako i negativne sopstvene vrednosti, dakle ona je nedefinitna, to se ovaj zadatak svodi na nekonveksnu optimizaciju. Kako se graf G ne može smatrati praznim, to ne umanjuje opštost da je ${a_{12}}^G = 1$. Iz toga se vidi da je glavni minor matrice A negativan.
Izvesno je da se bilo koja nedefinitna matrica može predstaviti kao razlika dve pozitivno definitne matrice:

$$A = A_1 - A_2, A_1, A_2 > 0$$

pa se 5.3 može zapisati kao

$$F(x) = f(x) - g(x) \uparrow \max, x \in D$$

tako da se za funkciju kreira razlaganje preko dve konveksne funkcije (difference convex, d.c.).
Neka je $d_i = \sum_{j=1}^{n} {a_{ij}}^G$ stepen čvora i u grafu G, važi $1 \leq d_i \leq n-2, i = 1, ..., n$. Smatra se da ne postoje izolovani čvorovi u grafu G.

Na osnovu razvoja kvadratne funkcije dobija se:

$$F(x) = \frac{1}{2}x^T Ax = \frac{1}{4}\sum_{i=1}^{n} {x_i}^2 + \sum_{(i,j)\in E} x_i x_j \tag{5.4}$$

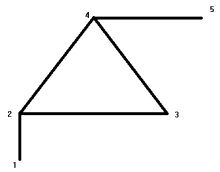
Dalje važi

$$f(x) = \frac{1}{4}(\sum_{i=1}^{n} {x_i}^2 + 2\sum_{(i,j)\in E}(x_i + x_j)^2)$$

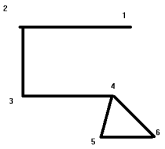
i

$$g(x) = \frac{1}{2}\sum_{i=1}^{n} d_i {x_i}^2$$

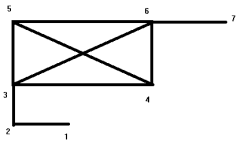
Funkcije $f(x)$ i $g(x)$ su strogo konveksne, tako da se ovim postupkom problem svodi na konveksnu optimizaciju.



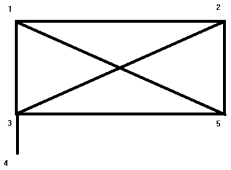
(a) 5 čvorova, veličina klike 3, G5_3



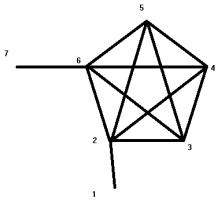
(b) 6 čvorova, veličina klike 3, G6_3



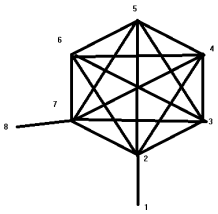
(c) 7 čvorova, veličina klike 4, G7_4



(d) 5 čvorova, veličina klike 4, G5_4



(e) 7 čvorova, veličina klike 5, G7_5



(f) 8 čvorova, veličina klike 6, G8_6

Slika 5.1: Grafovi

Predstavljene vrednosti funkcije cilja za klike koje su korišćene prilikom testiranja:

Ove vrednosti se uporedjuju sa dobijenim vrednostima u svakom algoritmu koji je testiran. Grafovi koji služe za testiranje se nalaze na Slici 5.1.

5.3.1 Algoritam Evolutivne strategije

Evolutivna strategija (ES) je optimizaciona tehnika bazirana na idejama adaptacije i evolucije [130]. Pripada generalnoj klasi veštačkih evolutivnih metodologija i evolutivnom izračunavanju. Kreirali su je Rechenberg i Schwefel [130] 60-tih godina prošlog veka. Evolutivna strategija

Tabela 5.5: Globalni minimum za prva četiri grafa koja su testirana:

Graf	Veličina klike	Vrednost funkcije cilja
G5_3	3	0.41666
G7_4	4	0.437500
G7_5	5	0.450000
G8_6	6	0.458333

koristi prirodne procese, primarno mutaciju i selekciju kao operatore pretrage. Kao i sa svim evolutivnim algoritmima, operatori se primenjuju u petlji. Iteracije u petlji se nazivaju generacije. Sekvenca generacija se ponavlja dok se ne dostigne kriterijum zaustavljanja. Sve dokle postoji realni prostor pretrage, mutacija se izvodi dodavanjem normalno rasporedjene slučajne vrednosti svakoj vektorskoj komponenti. Veličina koraka mutacije je standardna devijacija normalne distribucije i često je vodjena samo-adaptacijom (evoulutivni prozor) ili kovarijansnom matričnom adaptacijom (CMA-ES). Prirodna selekcija u evolutivnim strategijama je odredjena i bazirana na rangiranju fitovanih vrednosti, a ne pravih vrednosti. Rezultujući algoritam je invarijanta u odnosu na monotonu transformaciju funkcije cilja. Evolutivna strategija koja je primenjena na 5.3 operiše na populaciji veličine dva koju čine trenutna tačka i rezultat njene mutacije. Mutant postaje roditelj sledeće generacije ukoliko je fitovana vrednost funkcije za njega veća od fitovane vrednosti funkcije njegovog roditelja. U drugom slučaju mutant se odbacuje. Ovaj tip ES se naziva $(1+1)$ES. Generalno, mutanti se generišu i takmiče sa roditeljima. U nekim slučajevima postoje dokazi linearne konvergencije za unimodalne funkcije cilja. Metoda ES potpada pod metode takozvanog nultog reda. Postoje tri vrste metode u zavisnosti od stepena korišćenja derivativa, pa tako metoda prvog reda koristi gradient (prvi derivativ), metoda drugog reda koristi Hesian (drugi derivativ), dok se nultim redom označava metoda koja je "derivative-free" ili direktna metoda pretrage. Ova metoda aproksimira gradient a time i metodu prvog reda. Ako f kao funkcija cilja ima karakteristiku da izvod ne može da se dobije, u slučaju kada gradienti nisu dostupni tada je izvesno koristiti metodu nultog reda. Kod kontinualne optimizacije situacija je drugačija. Mnogi rezultati su dobijeni na osnovu empirijskih istraživanja, karakteristike konvergencije su dobijene na osnovu pojednostavljivanja modela stohastičkih procesa i preko postavljanja broja dimenzija na beskonačnost. Ipak, svi rezultati ovog tipa se baziraju na eksperimentalnim evaluacijama. Najuspešniji pristup do sada je primer koji uzima u obzir adaptaciju mutacije u svakom koraku. U okviru algoritma $(1+1)$ ES koristi se pravilo $\frac{1}{5}$ [130]. Algoritam u slučaju maksimalne klike je implementiran na osnovu datog opšteg algoritma Evolutivne strategije $(1+1)$ zbog toga što zahteva glatku funkciju cilja na koju je mogla da se primeni Metoda penal funkcije (engl. Exterior point method). Funkcija cilja je predstavljena kao zbir funkcije cilja sa ograničenjima i funkcije penala. Funkcija penala je data sa

$$P(x) = \frac{1}{\mu}\Big(\sum_{i=1}^{n} x_i - 1\Big)^2$$

Fitovanje funkcije uzima u obzir prostor funkcija koji se predstavlja preko funkcije sfere (engl. Sphere-function) a koji glasi: $Sphere(x) = \sum_{i=1}^{n} {x_i}^2 = x^T Ix$. Za sfernu funkciju cilja u radu [11] je pokazano da pravilo $\frac{1}{5}$ (u slučaju implementacije algoritma, pomera se σ za $\frac{4}{5}$) polovi rastojanje od početne tačke x do optimalnog rešenja u $\Theta(n)$ koraka i da je to asimptotski najbolje moguće tj. da je za svaku izotropnu mutaciju broj očekivanih izračunavanja funkcije f jednak $\Omega(n)$.

Jednostavan EA koristi mutaciju u odnosu na individualnu populaciju, gde je "individualna" sinonim za tačku pretrage. Neka je $c \in R^n$ trenutna individua. Data je početna tačka, na koju je postavljena promenljiva c, tada $(1+1)$ES izvodi sledeću petlju evolucije:
- Bira se vektor slučajne mutacije $m \in R^n$, gde je distribucija m zavisna od kursa optimizacionog procesa
- Generiše se mutant $c' \in R^n, c' = c + m$
- Ako je $f(c') \leq f(c)$ onda c' postaje trenutna individua $c = c'$ inače c' se odbacuje
- Ako je uslov zaustavljanja ispunjen onda je izlaz c inače vrati se na prvi korak

Kako se najgori mutant (s obzirom na funkciju koja se minimizuje) uvek odbacuje, $(1+1)$ES je slučajno penjanje na brdo (engl. Hill climbing), a pravilo selekcije se naziva elitistička selekcija. Originalno, vektor mutacije $m \in R^n$ se generiše preko generisanja komponenata vektora Gausove mutacije $\bar{m} \in R^n$ za koje se nezavisno postavlja standardna normalna distribucija, a zatim se vektor skalira preko multiplikacije skalarom $s \in R_{>0}, m = s*\bar{m}$. Gausove mutacije su najtipičnije mutacije u prostoru pretrage R^n. Neka je $|x| \in R^n$ euklidova dužina vektora $x \in R^n$, odnosno L_2 norma. Glavna karakteristika Gausove mutacije je da su $\bar{m}$ i m izotropno rasporedjene, odnosno $m/|m|$ je uniformno rasporedjeno u jediničnoj hiper-sferi i dužina mutacije je slučajna promenljiva $|m|$ nezavisna od pravca $m/|m|$. Pitanje je kako izabrati faktor skaliranja s. Što je manja greška aproksimacije, c je bliže tački optimuma. Na osnovu eksperimenata i grube kalkulacije predložen je parametar u iznosu od $\frac{1}{5}$ za uspešno pravilo. Ideja koja stoji iza ovog je da mutant treba biti prihvaćen sa verovatnoćom od $\frac{1}{5}$. Mutacija koja rezultuje $f(c') \leq f(c)$ se naziva uspešnom i tada je uspešna verovatnoća ona za koju je mutant $c' = c + m$ dobar makar kao c. Kada je elitistička selekcija iskorišćena uspešna verovatnoća koraka je kada je mutacija prihvaćena. Ako je svaki korak uspešan sa verovatnoćom $\frac{1}{5}$, smatra se da je prosek svake pete mutacije uspešan. Dakle, pravilo $\frac{1}{5}$ podrazumeva da se optimizacioni proces izvodi u n koraka, bez promene s.

Ako je jedan od pet koraka uspešan u observaciji, s se duplira, inače se množi brojem $\frac{8}{10}$. Početak testiranja se svodio na parametre 2 i 1/2 ali su se kasnije u zavisnosti od eksperimenata menjali faktori. Ovim postupkom se zadržalo pravilo $\frac{1}{5}$ potrebno da se vrednost parametra s mnogo ne promeni, prilikom više iteracija alogirtma. Sa a je obeležen koeficijent sa kojim se množio s ukoliko je korak bio uspešan, a sa b se obeležio koeficijent sa kojim se množio s ukoliko korak nije bio uspešan. Približno je testirano da važi

$$a * (b^5) \approx 1$$

ako je a jednako 2 onda je b trebalo da bude približno $\frac{1}{2}^{1/5}$ što iznosi 0.87055056. Zbog toga je kao zaokružena vrednost uzeta $b = \frac{8}{10}$.

Algoritam 19 Algoritam evolutivne strategije(1+1)

Početna tačka je postavljena u x^*
$f^* \leftarrow f(x^*)$, f je funkcija cilja
$s \leftarrow 1, i \leftarrow 0$
repeat
  Generisati slučajnu tačku x sa raspodelom $N(x^*, s)$
  $v = f(x)$
  if $v > f^*$ **then**
    $f^* \leftarrow v, x^* \leftarrow x$
    $s \leftarrow s * 2$
  else
    $s \leftarrow s * 0.8$
  end if
  $i \leftarrow i + 1$
until $i < maxiter$
return x^*

5.3.2 Algoritam GLOB

Lokalni optimizatori koji su primenjeni na MC problem

Postoje četiri izgradjene heuristike u trenutnoj verziji paketa koje su korišćene u testiranju MC problema na 5.3: Monte Karlo (slučajna pretraga), Multistart lokalna pretraga (MLS),

VNS i GausVNS (GaVNS). U svim osim u prvom, nove slučajne tačke su inicijalne tačke za izabran lokalni minimizator. U okviru MLS metode nova slučajna tačka se kreira na osnovu uniformne distribucije u čitavom boks regionu. U VNS nova slučajna tačka se bira iz skupova okolina najbolje izabrane optimalne tačke. U GausVNS metodi nova slučajna tačka se bira na osnovu dodatog gausovog šuma, odnosno iz normalne raspodela koja je odredjena standardnom devijacijom. Postoji nekoliko parametara koji definišu tip okolina i slučajnu distribuciju (metriku) koja se koristi za sledeću slučajnu tačku. Lokalni minimizatori koji se koriste u GausVNS, VNS i MLS metodama su vrlo poznati u teoriji nelinearnog programiranja: Nelder-Mead(NM), Hooke-Jeeves(HJ), Rosenbrock(RO), Steepest Descent(SD), Fletcher-Powell(FP) i Fletcher-Reeves(FR). Prve tri metode ne koriste gradientnu pretragu i koriste se kod neglatkih funkcija cilja. Druge tri metode koriste informacije o gradientu i on se računa direktno ili koristeći aproksimaciju konačnih razlika. Mnogi lokalni minimizatori koriste jednodimenzioni optimizator koji se oslanja na Pretragu zlatnog preseka (Golden section search) i na Metodu kvadratnih aproksimacija (Quadratic approximation method).

GLOB opcije primenjene na MC problem

Prva grupa parametara opisuje problem optimizacije. Postavljajući *function_name* korisnik može da izabere jednu od baznih testnih funkcija ili da definiše ime za svoju funkciju. Ukoliko funkcija zavisi od nekog fiksnog parametra, broj parametara se može promeniti u opciji *no_fun_params* a njihove vrednosti u *fun_params*. Ove vrednosti se smeštaju u globalni red *Fun_Params*[] i koriste se iz poziva korisničke ili testne funkcije. Vrednosti ograničenja se postavljaju u *left_boundaries* i *right_boundaries* uz postavljanje i reda vrednosti. Inicijalna tačka se postavlja u parametru *initial_point*. Ukoliko su sva ograničenja i inicijalne tačke iste vrednosti oni se mogu napisati u *left_boundaries_all*, *right_boundaries_all* ili *initial_point_all*. Ovo je vrlo korisno za testne probleme sa velikom dimenzijom prostora. Ukoliko korisnik zna koliki je tačni globalni minimum može ga postaviti u *glob_known_minimum* i kasnije se može informisati u izveštaju o procentu razlike izmedju trenutne i optimalne vrednosti funkcije. Parametar *boundary_tol* se koristi za identifikaciju postignute granice u metodi lokalne optimizacije. Gradientne metode se menjaju u metodu projektovanog gradienta kada tačke dostižu granicu.

Vrednost *significant_fun_difference* se koristi da proceni vrednost najmanjeg poboljšanja u funkciji (oko istog lokalnog minimuma) od značajnijeg i koristi se samo u statističkom izveštavanju.

Program u jednom izvršavanju optimizacije pokušava da nadje bolji minimum sve dok vreme postavljeno *glob_time_limit* parametrom ne prodje, ili broj iteracija u *glob_max_iteration* se ne postigne.

Za pronalaženje prosečnog i najboljeg rezultata u broju ponovljenih job izvršavanja korisnik može postaviti *glob_job_repetitions* izvršavanja kako bi se kod izvršavao automatski sa pridruženim statističkim izveštajem.

Program sadrži sopstveni uniformni generator slučajnih brojeva. Seed za ovaj generator se postavlja u *random_seed*. Ako je postavljen na 0, seed postavlja sistemski sat. Za VNS metaheuristiku, paket može koristiti tri različita tipa struktura okolina. Jedan tip je L_∞ sfera sa uniformnom distribucijom u njoj. Drugi i treći su L_1 sfere sa uniformnom i specijalno dizajniranom distribucijom u njoj. Postavljanje $random\_distribution = 111$ znači da se tri tipa distribucija menjaju ciklično dok postavljanje na 100 znači da se samo prvi tip strukture okolina koristi.

Sledeća grupa parametara definiše VNS strukturu okolina: $vns\_k_{\max}$ je broj sfera različitih veličina centriranih u trenutnoj tački. Veličinu sfera definiše korisnik ili veličina može biti automatski generisana softverom. Tri parametra *random_from_disk*, *reject_on_return* i *reject_on_return_k_diff* kontrolišu postavljanje slučajne tačke i služe kao prevencija povratka u trenutni lokalni minimum. Na primer, ako korisnik postavi *reject_on_return* na 1, kod bilo koje lokalne optimizacije koja počinje od N_k, okolina će biti prekinuta ukoliko se tačka vrati u N_{k-m} okolinu, gde je m postavljen preko *reject_on_return_k_diff*. Ukoliko je $random\_from\_disk = 1$ onda se slučajna tačka bira iz diska N_k/N_{k-1}. Osam parametara sa prefiksom *rep* kontrolišu izveštavanje koje je opisano sa više detalja u sledećem delu. Ostatak parametara kontroliše lokalni minimizator koji se koristi u svakoj metaiteraciji. Lokalni minimizator se prekida ako se nešto od sledećeg desi: broj iteracija dostigne maksimum, tačke dobijene na osnovu dve konsekutivne iteracije su bliže nego *ls_eps*, vrednosti funkcije dobijene

na osnovu dve konsekutivne iteracije su bliže nego ls_fun_eps ili je norma gradienta manja nego ls_grad_eps.

Statistički izveštaji

U jednom izvršavanju programa mogu se izvršavati jedan ili više job-ova. Ukoliko se izvršava više job-ova, u izveštaju se ispisuju najbolja i prosečna optimalna vrednost. Korisnik bira nivo izveštavanja tako što uključuje ili isključuje izvesne informacije. Korisnik dobija informaciju svake metaiteracije i svake uspešne metaiteracije. U toj informaciji se nalaze podaci o koordinatama slučajne tačke, o koordinatama najbolje tačke u lokalnoj optimizaciji, podaci o rekalkulisanoj vrednosti radiusa za VNS, kao i o nekim drugim statistikama. Na kraju job-a prikazuju se rezultati o broju metaiteracija, proteklom vremenu, učinku računanja, itd. Poslednja značajna metaiteracija je ona posle koje nijedno značajno poboljšanje funkcije ne može da se postigne. Ovaj koncept prepoznaje mala poboljšanja u istom najboljem lokalnom minimumu nakon momenta kada je taj lokalni minimum praktično pronadjen. Za VNS metaheuristiku broj uspešnih koraka se prikazuje za svaku okolinu u kojoj se bolja vrednost funkcije pronalazi. Prikazuje se i statistika broja uspešnih koraka za različite slučajne distribucije (metrike) koje se koriste kako bi se pronašla nova početna tačka. Ukoliko korisnik zna koja je tačna vrednost globalnog minimuma može ga postaviti u fajl gde se nalaze parametri, a nakon izvršavanja može pročitati koliki je procenat odstupanja poznate od trenutno pronadjene najbolje vrednosti funkcije cilja preko formule $(f - f_{best})/f_{best} \times 100$ za svaku prikazanu iteraciju.

Izveštaj za MC problem

Kako je GLOB kreiran prvenstveno za probleme globalne optimizacije sa velikim brojem lokalnih minimuma, pogodan je za primenu na razmatrani problem maksimalne klike. Klika se prikazuje preko Jednačine 5.4. Korišćene su selektovane heurističke metode slučajne tačke, MLS i VNS. Testirana je lokalna pretraga u odnosu na Steepest descent, Fletcher-Powell, Fletcher-Reeves, Nelder-Mead, Hook-Jeeves, Rosenbrock i Minmax fun. Selektovana je metoda konačnih razlika za izračunavanje gradijenta. Metoda linearne pretrage koja je testirana je zlatni presek. Generisani su radiusi izmedju centara okolina rešenja. Parametar najveće dimenzije skupa promenljivih je podešavan na relativno mali (do 250) s obzirom da broj promenljivih utiče na brzinu i optimizaciju. Pokazalo se da opcija VNS i lokalnog optimizatora koji koristi Nelder-Mead metodu daje najbolje rezultate, što se i očekivalo s obzirom da pomenuta metoda ne zahteva postojanje i poznavanje parcijalnih izvoda funkcije.

5.3.3 Algoritam Generalne metode promenljivih okolina

Ovaj algoritam je nastao na osnovu paketa GLOB i predstavlja heuristiku za rešavanje problema kontinualne optimizacije bez ograničenja i sa ograničenjima [175]. Baziran je na metodi Generalne promene okolina. Koriste se različite okoline i distribucije, indukovane iz različitih metrika koje su rangirane i koje se koriste za izbor slučajnih tačaka u koraku razmrdavanja. Na kontinualnu formulaciju MC 5.3 se primenjuje algoritam koji se sastoji od funkcije penala spoljašnje tačke i kombinacije sekvencijalne i egzaktne penal transformacije.

Metoda penala spada u klasične pristupe za nalaženje lokalnih minimuma 5.4. Ove metode rešavaju probleme optimizacije sa ograničenjima, tako što rešavaju sekvence problema bez ograničenja. Problemi bez ograničenja uključuju pomoćne funkcije koje se sastoje od funkcije cilja i uslova penala koji mere ispunjenost ograničenja. Metoda penala uključuje dve velike grupe: metoda penala spoljašnje tačke koja uključuje penaliziranje ispunjenosti penala i metoda penala unutrašnje tačke koja uključuje penal za dostizanje granice kod ograničenja tipa nejednakosti. Kako Model 5.4 ima i ograničenje tipa jednakosti i ograničenje tipa nejednakosti, prirodno je da se koristi metoda spoljašnje tačke. Koristi se ta tehnika penala lokalne minimizacije funkcije $-f$ i kombinuje se sa VNS strategijom kako bi se dostiglo rešenje problema blisko optimalnom. Odgovarajući problem bez ograničenja minimizira takozvanu funkciju penala i ima oblik:

$$\min_{a \le x \le b} F_{\mu,q}(x) = f(x) + \frac{1}{\mu} P_q(x) \tag{5.5}$$

uz penal:

$$P_q(x) = \sum_{i=1}^{m} (\max\{0, g_i(x)\})^q + \sum_{i=1}^{r} |h_i(x)|^q$$

gde je μ pozitivni parametar penala i $q \geq 1$ je eksponent penala. Ako je $q = 1$ može se dokazati da je funkcija penala egzaktna, za neku dovoljno malu vrednost μ lokalnog rešenja problema koji je lokalni minimum 5.5. Funkcija egzaktnog penala nije diferencijabilna u svim tačkama. Zbog toga, vrednosti $q > 1$ garantuju diferencijabilnost i to se koristi. U ovom slučaju rešenje 5.5 problema pod blažim uslovima konvergira lokalnom minimumu kada $\mu \to 0$ [14]. Glavno dostignuće Metode kontinualnih generalnih promenljivih okolina (engl. CGVNS) je primena VNS metodologije na 5.5 uz varijaciju parametra penala. Pronadjeno je inicijalno rešenje x u inicijalnom koraku kao dopustivo rešenje 5.5 za fiksni eksponent penala q i μ_0, gde je sa μ_0 data inicijalna vrednost parametra penala μ. Tačka y u koraku razmrdavanja se slučajno generiše kao dopustivo rešenje 5.5 za trenutnu vrednost μ, a u koraku lokalne pretrage, metod se primenjuje kako bi se pronašao lokalni minimum y' ovog problema. Ako je y' bolje od x u funkciji $F_{\mu,q}$ algoritam se pomera u y' i nastavlja se sa istim μ. Inače, algoritam ostaje u x i vrednost od μ opada. Ako je μ veće od nekog datog minimalnog μ_{min}, μ se ažurira množeći ga sa konstantnim faktorom $\alpha, 0 < \alpha < 1$ i funkcija penala se menja. Za svaku tačku y' koja se generiše procedurom lokalne pretrage, odgovarajući penal se izračunava. Ukoliko nije veći od nekog malog dopustivog faktora tolerancije ϵ, tačka se smatra dopustivom i kao takva se pamti.
Dodatni parametri su:
q - eksponent funkcije penala
$\alpha \in (0, 1)$ - rata pada penala
ϵ - mali broj, tolerancija za proveru dopustivosti
μ_0, μ_{min} - inicijalna i minimalna vrednost parametra penala respektivno
Postavljeni su parametri na $q = 2$, $\alpha = 0.9$, $\epsilon = 10^{-10}$, $\mu_0 = 1$ i $\mu_{min} = 10^{-8}$. Parametri su inicijalno tako postavljeni, ali su u zavisnosti od primera do primera prilagodjavani i menjani.

Algoritam 20 GVNS

Inicijalizacija. Selektuj skup strukture okolina $N_k, k = 1, ..., k_{\max}$, parametre funkcije penala q, μ_0, μ_{min} i red tipova slučajnih distribucija;
Izaberi slučajnu inicijalnu tačku x koja je dopustiva za probleme sa boks ograničenjima
Postavi $x^* \leftarrow x, f^* \leftarrow F_{\mu,q}(x), {f^*}_F \leftarrow \infty, \mu \leftarrow \mu_0$
repeat
  Postavi $k \leftarrow 1$
  repeat
    Za sve distribucije iz reda generiši slučajnu tačku $y \in N_k(x^*)$
    Primeni neku metodu lokalne pretrage od y kako bi se dobio lokalni minimum y'
    if $P_{\mu,q}(y') \leq \epsilon$ i $f(y') < {f^*}_F$ **then**
      ${x^*}_F \leftarrow y', {f^*}_F \leftarrow f(y')$
      if Ako je $F_{\mu,q}(y') < F_{\mu,q}(x^*)$ **then**
        $x^* \leftarrow y'$ i postavi k na 1
      else
        Postavi $\mu \leftarrow \max(\mu_{min}, \alpha\mu)$
        Postavi $k \leftarrow k + 1$
      end if
    end if
  until $k > k_{\max}$
  if ${f^*}_F = \infty$ **then**
    Dopustivo rešenje ${x^*}_F$ se dobija preko lokalnog minimizatora koji je primenjen na $F_{\mu,q}(x)$ sa $q = 1, \mu = \mu_{min}$ počev od x^*
  end if
until Kriterijumi za prekid nisu ispunjeni
Tačka ${x^*}_F$ je aproksimativno dopustivo rešenje problema

CGVNS implementira strukture okolina koje su definisane metrikama L_1 i L_∞ i sa četiri vrste distribucija. Distribucije su formirane na sledeći način:
- D_1 je distribucija generisana specijalnom distribucijom u L_∞ jediničnoj sferi
- D_2 je distribucija generisana specijalnom distribucijom u L_1 jediničnoj sferi
- D_3 je distribucija generisana specijalno dizajniranom distribucijom u L_1 jediničnoj sferi tako

što se svaka koordinata odredjuje uniformno na sukcesivnim odsečcima
- D_4 je distribucija koja je na početku generisana specijalnom distribucijom u L_∞ jediničnoj sferi, a zatim svaka tačka x i boks ograničeni vektori a i b su skalirani preko faktora $(b_i - x_i)/r_{k_{\max}}$. Suprotan znak se uzima ukoliko su kooridnate manje od nule. $N_{k_{\max}}(x)$ je jednak celom boks regionu S i $N_k(x)$ je boks region S koji je skupljen oko tačke x sa faktorom skupljanja $r_k/r_{k_{\max}}$. Broj $k_{\max}$ različitih sfera u jednoj metrici je ulazni parametar. Veličina sfera se eksplicitno definiše a može i automatski da se podesi tako što kad se pronadje najveća distanca R od trenutne tačke x do granice boks regiona, radius se kalkuliše kao $\frac{kR}{K_{\max}}$. Slučajna tačka y u koraku razmrdavanja se može izabrati iz diska tako što zadovoljava dodatni zahtev $y \in N_k(x) \setminus N_{k-1}(x)$. Inicijalno rešenje postavlja korisnik proizvoljno. Kriterijumi zaustavljanja se takodje definišu od strane korisnika, a najčešće su to CPU vreme $t_{\max}$ ili maksimalan broj VNS iteracija. Ukoliko je plan da se redukuje broj parametara, CGVNS koristi skup parametara koji daju dobar prosek izvodjenja u najviše eksperimenata, a to je $k_{\max} = 15$, sekvenca struktura okolina u L_∞ i sva četiri tipa distribucije (D_1,D_2,D_3,D_4). Pravac najbržeg spusta je odredjen gradijentom funkcije cilja a kasnije se pretraga obavlja po liniji (engl. Line search) kako bi se pronašla optimalna tačka na pravcu.

5.3.4 Generalizovani Motzkin-Straus algoritam

Algoritam [75] je primenjen na metodu FSS kao na skup nekonveksnih formulacija. Kreiran je Generalizovan Motzkin-Straus algoritam, a kroz to ispitivano da li je moguće obuhvatiti sve različite formulacije. Ideja je bila da se uradi pretraga kroz prostor rešenja ali i kroz prostor formulacija. Ispitivano je koja su rešenja $1-1$ preslikavanja u odnosu na formulaciju, a koja rešenja pripadaju različitim formulacijama. Suština je u tome da se pretraga i razmrdavanje kreću kroz prostor formulacija i da se ne zadržavaju u lošijim rešenjima, tzv. zamkama. Ukoliko se to desi u okviru neke formulacije, uvek postoji mogućnost da se pretraga i razmrdavanje vrše u okviru sledeće formulacije. Istraživanje FSS problema maksimalne klike podrazumeva uvodjenje skupa različitih formulacija zbog potrebe pronalaženja optimalnog rešenja klike [172]. Svaka formulacija predstavlja se skupom okolina i funkcijom cilja $\mathcal{F}(N_l(\phi), f(x)), x \in X, l \in 1, ..., l_{\max}$, $\phi \in \mathcal{F}$, a pretraga se kreće od jedne formulacije do druge, u zavisnosti da li je rešenje lokalni optimum ili stacionarna tačka trenutne formulacije. Pored pronalaženja stacionarnih tačaka pronalaze se i veze izmedju optimalnih rešenja raznih formulacija. Svaki put kad algoritam dodje do stacionarne tačke, problem se nelinearno transformiše, ali bez garancija da će se dobiti lokalni minimum. Jedna od varijanti koja je testirana je varijanta koja ne menja formulacije (l okoline) oko fiksnog rešenja x, već menja rešenja x u okviru formulacije. Gibbons i drugi [93] su proširili svoj rezultat tako što su predvideli karakterizaciju maksimalne klike u smislu lokalnih rešenja. Proučavali su uslove optimalnosti Motzkin-Straus programa i razmatrali karakteristike predstavljene parametrizacije odgovarajućeg kvadratnog programa. Gibbons i drugi [93] su predložili drugu neprekidnu formulaciju problema maksimalno stabilnog skupa. Podsećanja radi, Motzkin-Straus su razmatrali optimizaciju kvadratne funkcije sa L_1 ograničenjima.

Teorema 5.3.1 *Razmatra se sledeći optimizacioni problem: Optimalna vrednost za sledeći kvadratni program koja se istražuje je*

$$V(k) = \min_x (\frac{1}{2} x^T A_G x + (\sum_{i=1}^{n} x_i - 1)^2) \tag{5.6}$$

p.o.

$$\sum_{i=1}^{n} {x_i}^2 \leq \frac{1}{k}$$

$$x \geq 0$$

Ako je $\bar{x}$ rešenje 5.6 onda je $V(k) = 0$ akko postoji stabilan skup I u G tako da je $|I| \geq k$.

Usvojena je ideja korišćenja optimizacije iste kvadratne funkcije na L_p sferi kako bi se pronašao veći stabilni skup, samo što se uz to koristila informacija o svim stacionarnim tačkama 5.6. Korišćeni su računarski rezultati Gibbons-a i drugih [93] kako bi se izračunala efikasnost pristupa.

Vrlo je jednostavan da se implementira, sa vremenom složenosti $O(|E|)$, gde je $|E|$ broj ivica. Algoritam je testiran na grafovima malih dimenzija sa tendencijom da se u budućnosti testiranje proširi na analizu velikih baza podataka.
Neka je $x^{(0)} = (1...1)^T$ i neka je $\beta = 1.1$. Neka je $A = \{0,1\}^{n\times n}$ matrica susedstva netežinskog, neusmerenog grafa. Neka je $A_{ii} = 1, 1 \leq i \leq n$. Algoritam iterativno ažurira x koristeći:

$$x_i{}^{(t+1)} = (x_i{}^{(t)} \frac{(Ax^{(t)})_i}{[x^{(t)}]^T Ax^{(t)}})^{1/\beta} \tag{5.7}$$

Neka je $x^* = (x_1{}^*, ..., x_n{}^*)^T$ konvergentno rešenje i neka je C podskup koji odgovara nenula elementima u x: $C = \{i|x_i{}^* > 0\}$. Tada je C maksimalna klika. Motzkin-Straus formula kao kvadratna formula zapisana na sledeći način glasi:

$$\max_x x^T Ax, p.o. \qquad \sum_{i=1}^{n} x_i = 1, x \geq 0, diag(A) = 0 \tag{5.8}$$

Teorema 5.3.2 *Neka je G netežinski graf i neka je x^* optimalno rešenje za 5.8. Neka je $C = \{i|x_i{}^*\}$ podskup koji odgovara nenula elementima. Ako nenula elementi imaju istu vrednost $x_i{}^* = 1/|C|, \forall i \in C$, vektor x^* se naziva karakterističan vektor podskupa C gde je C maksimalna klika u G.*

Ono što je otežavajuće za izračunavanje maksimalne klike jeste situacija da je matrica A nedefinitna i $diag(A) = 0$. Generalizacija Motzkin-Straus (GMS) teoreme je motivisana činjenicom da je $diag(A) = 0$, odnosno da su sopstvene vrednosti A ravnomerno podeljene sa obe strane nule:

$$\sum_{k=1}^{n} \lambda_k(A) = TrA = 0$$

Postoji mnogo lokalnih optimalnih rešenja, i svako od njih odgovara maksimalnoj kliki. Globalno rešenje koje odgovara maksimalnoj kliki je teško naći. Matrica A zapisana je kao

$$\bar{A} = A + I$$

i na osnovu toga rešava se kvadratni problem od $x^T \bar{A} x$. Ovo je lakši oblik problema da se reši zbog

$$\sum_{k=1}^{n} \lambda_k(\bar{A}) = n$$

odnosno postoji više (jačih) sopstvenih vektora sa pozitivnim sopstvenim vrednostima i manje (slabijih) sopstvenih vektora sa negativnim sopstvenim vrednostima. Ukoliko je $\bar{A}$ pozitivno definitna postoji jedinstveno globalno rešenje, problem je konveksan iako je je funkcija kvadratna. Ipak, generalno, $\bar{A}$ nije pozitivno definitna. Rešavanje problema kvadratne optimizacije sa novom matricom je lakše nego sa originalnom matricom. Prava generalizacija

$$\max_x x^T \bar{A} x, \sum_{i=1}^{n} x_i = 1, x \geq 0$$

nije validna generalizacija zato što u ovom formalizmu za rešenje nije garantovano da konvergira maksimalnoj kliki. Ova poteškoća je rešena modifikovanjem ograničenja, odnosno L_1 norme. Umesto nje, ograničenje se zapisuje kao L_p norma. Za vektor x, L_p norma je $\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$. Dokle god je $p \simeq 1$, vektor optimalnog rešenja je redak, odnosno mnogi elementi u x^* su nule. Koristeći L_1 normu ograničenja sprovodi se retkost. Zbog promene u normi, formula se zapisuje u konačnom obliku

$$\max_x x^T \bar{A} x$$

p.o.

$$\sum_{i=1}^{n} x_i{}^{\beta} = 1, x \geq 0, \beta \in [1,2] \tag{5.9}$$

Kao što se vidi, uveden je parametar β za koga se pokazuje da je oblika $\beta = 1+\epsilon, 0 < \epsilon \ll 1$. Za takvo β dokazuje se konvergencija i demonstrira se da ako je $\beta \to 1_+$ retkost u okviru rešenja stalno raste, pokazujući blisku relaciju sa L_1 ograničenjem. Za $\beta = 2$ rešenje je dato preko glavnog sopstvenog vektora od A.
Generalizovana Motzkin-Straus teorema:

Teorema 5.3.3 *Neka je $\bar{A}$ matrica susedstva grafa, a $\beta = 1+\epsilon, 0 < \epsilon \ll 1$. Neka je $C = \{i|x_i{}^* > 0\}$ podskup koji odgovara nenula elementima rešenja. Ako nenula elementi imaju iste vrednosti, $x_i{}^* = 1/|C|^{1/\beta}, \forall i \in C$, tada je C maksimalna klika.*

Dokaz:
Dokaz objašnjava dopustivost, korektnost i konvergenciju rešenja.
Kako nenula elementi od x imaju konstantne vrednosti, x^* mora imati sledeću formu:

$$x^* = (1/|C|^{1/\beta})(1...1, 0...0)^T$$

pretpostavlja se bez gubitka opštosti da su čvorovi u C indeksirani na početku. Funkcija cilja postaje

$$J = (x)^T \bar{A} x = |C|^{2-2/\beta}$$

Kako je $\beta > 1$, važi $2 - 2/\beta > 0$ i $\max J$ postaje ekvivalentno sa $\max |C|$ Ako se koristi $\beta = 1$ onda je $J = 1$ nezavisno od $|C|$, odnosno, ne garantuje se da je moguće izračunati maksimalnu kliku. Formalno, za definiciju maksimalne klike, date Formulom 5.7 se pokazuje dopustivost, korektnost i konvergencija.

1. Dopustivost - Pokazuje se da iz bilo koje inicijalne tačke $x^{(0)}$ iteracija dovodi do dopustivog rešenja

$$\sum_i [x_i{}^{(t+1)}]^\beta = \sum_i \frac{x_i{}^{(t)}(Ax^{(t)})_i}{[x^{(t)}]^T A x^{(t)}} = 1$$

2. Korektnost - Pravilo ažuriranja prilikom konvergencije zadovoljava uslove Karuš-Kun-Takera (engl. Karush–Kuhn–Tucker, KKT). KKT pravilo podrazumeva da odredjeni uslovi prvog reda moraju biti zadovoljeni kako bi neka tačka x mogla biti rešenje nelinearnog optimizacionog problema. Neka je dat Model 5.3, gde je funkcija cilja glatka u celom domenu. Ako je x^* lokalni minimum koji zadovoljava uslove da su funkcije ograničenja afine, tada postoje konstante $\mu_i (i = 1, ..., m), \lambda_j (j = 1, ..., l)$ koje se nazivaju KKT množitelji i za koje važi:

$$\nabla f(x^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{l} \lambda_j \nabla h_j(x^*)$$

   Kako bi se uslovi KKT primenili na optimizacioni problem, formira se Lagranžova funkcija tako da je

$$L = x^T A x - \lambda(\sum_i x_i{}^\beta - 1)$$

   gde je λ Lagranžov množitelj koji prati L_p ograničenje. KKT uslovi za nenegativnost x_i, dovode do optimalnog rešenja x^* koje mora da zadovolji

$$[2(Ax)_i - \lambda\beta[x_i]^{\beta-1}]x_i = 0$$

   Nakon sumiranja po i dobija se vrednost za množitelj λ

$$2x^T A x = \lambda\beta \sum_{i=1}^{n} [x_i]^\beta = \lambda\beta$$

   Prilikom zamene vrednosti za λ jednačina postaje

$$[x_i{}^{(t+1)}]^\beta = x_i{}^{(t)} \frac{(Ax^{(t)})_i}{\lambda\beta/2}$$

3. Konvergencija - Pojašnjenje ove karakteristike je na sledeći način:

Teorema 5.3.4 *Na osnovu pravila o ažuriranju 5.7 iteracija konvergira ka fiksnoj tački.*

Dokaz:
Dokazuje se karakteristika monotonosti:

$$L(x^{(0)}) \leq L(x^{(1)}) \leq L(x^{(2)}) \leq ... \leq L(x^{(t)}) \leq ...$$

Uvodi se pomoćna funkcija $G(x, x')$ koja je pomoćna funkcija funkcije $L(x)$ ako je $G(x, x') \leq L(x)$; $G(x, x) = L(x)$. Sada se definiše

$$x^{(t+1)} = \arg\max_{x} G(x, x^{(t)}) \tag{5.10}$$

Po konstrukciji dobija se da je Lagranžijan jednak pomoćnoj funkciji u iteraciji t. Na osnovu gore uvedenih jednakosti (nejednakosti) dobija se:

$$L(x^{(t)}) = G(x^{(t)}, x^{(t)}) \leq G(x^{(t+1)}, x^{(t)}) \leq L(x^{(t+1)}).$$

Tako je $L(x^{(t)})$ monotono rastuća funkcija na osnovnu iterativnog pravila ažuriranja Formule 5.7. □

Sledeće se dokazuje da je pomoćna funkcija $G(x, x')$ konveksna funkcija odakle se dobija globalno optimalno rešenje. Funkcija se zapisuje kao:

$$G(x, x') = \sum_{ij} x'_i A_{ij} x'_j \Big(1 + \log \frac{x_i x_j}{x'_i x'_j}\Big) - \lambda\Big(\sum_i {x_i}^{\beta} - 1\Big)$$

Gradient (izvod prvog reda) je

$$\frac{\partial G(x, x')}{\partial x_i} = 2\frac{x'_i (Ax')_i}{x_i} - \lambda\beta {x_i}^{\beta-1} = 0$$

Hesian matrica (izvod drugog reda) je:

$$\frac{\partial^2 G(x, x')}{\partial x_i \partial x_j} = -\Big[2\frac{x'_i (Ax')_i}{{x_i}^2} + \lambda\beta(\beta - 1){x_i}^{\beta-2}\Big]\delta_{ij}$$

Kako je $\beta \geq 1$ Hesian je diagonalna matrica sa negativnim kvantitetima na dijagonalama, odnosno, to je konveksna funkcija. Postoji globalni optimum koji se dobija kad se gradient postavi na nulu. Na osnovu Formule 5.10 i postavljanja $x^{(t)} = x', x^{(t+1)} = x$ dobija se pravilo ažuriranja formule. □

FSS prati rastuće bliske okoline trenutnog rešenja i prelazi iz jedne formulacije u drugu. U okviru procedure koriste se različite okoline i izvode sistematske pretrage zbog čega se vodi računa o distanci bilo koja dva rešenja u okviru okolina. Okolina rešenja maksimuma klike predstavlja skup partitivnih skupova, odnosno skup različitih podskupova grafa.
Za svaku od formulacija se konstruiše prostor rešenja sa nekom metrikom odakle se indukuju okoline. U svakoj iteraciji se menja metrika tako da se smer pretrage prilagodjava bolje lokalnom obliku funkcije kroz strmo opadanje. U prvoj iteraciji se koristi jedinična sfera u n dimenzionom euklidskom prostoru a zatim se nalazi smer antigradijenta, odnosno najstrmijeg opadanja. U ostalim iteracijama koriste se elipsoidi i opet se izvodi najstrmije opadanje u odnosu na novu metriku koja je dobijena odredjenom linearnom transformacijom. Svrha algoritma je da se iterativno izgradi dobra aproksimacija inverza Hesijan matrice $A^{(-1)}$ funkcije f zbog opisa lokalne zakrivljenosti funkcije. Što se tiče strategije za inicijalno rešenje, uglavnom se razmatraju dve mogućnosti formiranja strategije za inicijalno rešenje: slučajno izabrano početno rešenje ili generisano konstruktivnom heuristikom gramzivog tipa. Poboljšanja i razmrdavanja podrazumevaju kako selektovati početnu formulaciju, kako dalje vršiti izbor formulacije, kako selektovati početnu okolinu i kako dalje menjati okoline i to su takodje pitanja koja će karakterisati istraživanje i u odnosu na njih planira se ekstenzija FSS algoritma. Glavni doprinos ovog pristupa jeste usvajanje metode više formulacija u okviru jednog izvršavanja algoritma, Generalizacijom Motzkin-Straus formalizma i korišćenjem L_p ograničenja. Izvršava se iterativni algoritam kako bi se računalo rešenje i dokazivala korektnost i konvergencija algoritma. Više formulacija olakšava računanje pseudo klika (tj., gusto povezanih podgrafova koji su manje gusti od klika).

Algoritam 21 GMS metoda za odredjivanje maksimalne klike

Izabrati početno rešenje x
Izabrati niz $\beta_l, \quad l = \{1, ..., l_{\max}\}$
Izabrati niz okolina $\phi_l, \quad l = \{1, ..., l_{\max}\}$
$l \leftarrow 1$
repeat
  Razmrdavanje: Izabrati slučajno $\beta \in \phi_l$
  repeat
    for $i = 1 \quad to \quad n$ **do**
      Lokalna pretraga
      $x_i \leftarrow (x_i \frac{(Ax)_i}{[x]^T Ax})^{1/\beta_l}$
    end for
  until x je dostigao stacionarnu tačku
  if $f(x) > f(x^*)$ **then**
    $x^* \leftarrow x$
    $l \leftarrow 1$
  else
    Razmrdavanje sa l-tom okolinom
    $l \leftarrow l + 1$
  end if
until Kriterijum zaustavljanja nije dostignut
return x^*

Prikaz rezultata prethodna četiri algoritama

Tabela 5.6: Testiranje na primerima malih dimenzija

Graf	Dim. grafa	Kardin. klike	ES	GLOB	GVNS	GMS
G5_3	5	3	4s	**<1ms**	<10ms	**<1ms**
G5_4	5	4	5s	**<1ms**	<10ms	**<1ms**
G6_3	6	3	4s	**<1ms**	<10ms	**<1ms**
G6_4	6	4	5s	**<1ms**	<10ms	**<1ms**
G7_4	7	4	5.18s	**<1ms**	<10ms	**<1ms**
G7_5	7	5	5.20s	**<1ms**	<10ms	**<1ms**
G8_3	8	3	6s	**<1ms**	<10ms	**<1ms**
G8_4	8	4	6s	**<1ms**	<10ms	**<1ms**
G8_6	8	6	6s	**<1ms**	<10ms	**<1ms**
prosek	**6.67**	**4**	5.15s	**<1ms**	<10ms	**<1ms**

Zajednička Tabela 5.6 predstavlja testiranje pojedinačnih i kombinovanih algoritama iz oblasti promena okolina i formulacija nad malim testnim grafovima čije su dimenzije date. Evolutivna strategija nije baš pogodna za dalji razvoj, s obzirom da je njeno vreme izvršavanja pet redova veličine više nego očekivano. Najbolje su se pokazali algoritmi GMS i GLOB. u okviru GMS se koristila promena formulacija, dok su se u okviru GLOB funkcionalno prilagodljivog paketa testirale ekstenzije VNS metode. GVNS takodje ima nešto duže vreme nego uobičajeni algoritmi. Metoda GMS je dala obećavajuće rezultate u okviru istraživanja FSS što ukazuje na to da se daljim poboljšanjima mogu postići bolji rezultati. Očekuje se da će nova metoda biti superiornija u odnosu na trenutne najbolje iz literature na grafovima većih dimenzija jer će se tada pokazati efekti promene formulacija ili okolina i formulacija. Svi pomenuti algoritmi dozvoljavaju da se promenom parametara ili grupa parametara dostigne bolja optimizacija. Takodje, moguće je definisati druge tipove okolina koje bi mogle da dovedu do boljeg rešenja u odnosu na publikovane u literaturi.

Na osnovu prethodna četiri algoritma i na osnovu [195] kreiran je algoritam FSS [139] koji je ostvario rezultate u okviru promena formulacija za problem maksimuma klike u grafu.

5.4 Metode promenljivih formulacija i stepena čvora višeg reda u rešavanju problema maksimuma klike

5.4.1 Uvod

U ovom odeljku predstavljeni su novi rezultati koji su dobijeni na osnovu istraživanja Enumerativne egzaktne metode kao i Metode pretrage prostora formulacija [137–139].
Kao što je već istaknuto, u literaturi postoje mnogi algoritmi koji su korisni prilikom rešavanja maksimuma i maksimalnih klika. Prilikom istraživanja Metode pretrage prostora formulacija primenjeno je nekoliko ekstenzija i posmatrano kako se rezultati ponašaju i menjaju u zavisnosti od promene logike algoritma. Kod izvesnog broja metoda vreme izvršavanja je bilo prihvatljivo za odredjene skupove grafova (što je prikazano u prethodnim odeljcima), dok drugi skup metoda u potpunosti nije bio u mogućnosti da izvede rezultate za velike grafove. U radu su prikazana tri algoritma, gde prvi predstavlja egzaktnu metodu rešavanja maksimuma klike, dok su drugi i treći Heurističke metode rešavanja maksimuma klike. Predstavljeni algoritmi su definisani u Uvodu i od sada se koriste skraćeni nazivi: FEHO, FHHO i FSSHO. Sva tri se baziraju na tehnici Redukcije kako bi brzo pronašli maksimum klike u velikim retkim grafovima. Primenjeni su obimni eksperimenti na različitim grupama i vrstama grafova, neki su sintetički kreirani, a neki su realni na osnovu dostupnih podataka. Predstavljeni rezultati pokazuju da predloženi algoritmi mogu biti i za čitav red veličine brži od do sada objavljenih algoritama. Metoda FSSHO koja je primenjena radi u pojedinim slučajevima i do red veličine brže od FEHO metode uz generisanje optimalnog ili rešenja vrlo bliskog optimalnom.
Na početku je razvijana FEHO metoda kako bi se proverilo koliko je efikasna u pretrazi klike maksimalne kardinalnosti. Radovi na temu algoritma grananja i ogradjivanja su poslužili kao početak istraživanja na temu tačnih metoda. Kako grananje sistematično pretražuje sve kandidate za rešenja, ogradjivanje sa druge strane predstavlja Redukciju, odnosno odbacuje loše kandidate na temelju prethodno izračunatih granica. Algoritam koji je napravljen se izvršavao na velikom skupu testnih grafova, a potom je uporedjivan sa algoritmima [51, 182, 195]. Smatra se da je FEHO za red veličine brži za velike retke grafove a da je vreme izvršavanja uporedivo za velike guste grafove. FHHO se izvršava nekoliko redova veličine brže od egzaktnog i daje rešenja koja su optimalna ili blizu optimalnih za većinu slučajeva. Predstavljene verzije algoritama su pogodne za paralelizaciju. Algoritmi koji su poslužili kao referentne tačke su dati u [51, 195] i predstavljeni su u odeljku o egzaktnim algoritmima.
FHHO i FSSHO su prikazani u okviru Algoritma 23 i Algoritma 24. FHHO predstavlja unapredjenu metodu heuristike u odnosu na [195], dok FSSHO predstavlja dostignuće u oblasti Metode promenljivih formulacija, odnosno dostignuće FSS.

5.4.2 Teorijski rezultati

Pre prikaza algoritama predstavljeni su originalni teorijski rezultati koji su korišćeni u njihovoj implementaciji.

Definicija 5.4.1 *Stepen reda j čvora V_i je definisan pomoću sume*

$$d_j(V_i) = \sum_{k \in N(V_i)} d_{j-1}(V_k),$$

za $j > 0$, $d_0(V_i) = 1$ i $N(V_i)$ je okolina čvora V_i.

Klasičan stepen čvora $d(V_i)$ je prema ovoj definiciji jednak stepenu prvog reda $d_1(V_i)$.

Lema 5.4.1 *Stepeni reda j čvorova podgrafa G' grafa G su manji ili jednaki od stepena čvorova grafa G istog reda.*

Dokaz se sprovodi indukcijom. Tvrdnja je tačna za nulti red. Ako se pretpostavi da je tačna za red j sledi na osnovu definicije da je

$$d'_{j+1}(V_i) = \sum_{k \in N'(V_i)} d'_j(V_k) \leq \sum_{k \in N(V_i)} d_j(V_k)$$

□

Lema 5.4.2 *Stepeni reda j čvorova kompletnog grafa sa C čvorova su dati izrazom*

$$d_j(V_i) = (C-1)^j$$

Tvrdnja je očigledno tačna za $j = 0$. Ako je tačna za red j sledi na osnovu definicije

$$d_{j+1}(V_i) = \sum_{k \in N(V_i)} d_j(V_k) = (C-1)(C-1)^j = (C-1)^{j+1}$$

□

što dokazuje lemu.
Kao direktna posledica prethodne dve leme dobija se teorema

Teorema 5.4.1 *Potreban uslov da bi neki čvor V_i bio čvor klike veličine N je da je stepen reda j tog čvora veći ili jednak od $(N-1)^j$ za svako $j \geq 0$.*

Ovaj uslov ($d_j(V_i) \geq (C-1)^j$) se može iskoristiti za strategiju Redukcije prilikom pretrage maksimuma klike što je u ovom radu iskorišćeno. Ovim su uvedeni strožiji kriterijumi za Redukciju nego što su u [195] i dodatno se popravio Carraghan-Pardalosov algoritam [51].

5.4.3 Brzi egzaktni algoritam višeg reda

Algoritam 22 [137] predstavlja poboljšanu verziju algoritma predloženog u [195] i kao i on baziran je na egzaktnom algoritmu predstavljenom u [51]. FEHO na sistematski način ispituje da li neki čvor grafa može biti u kliki koja je veća od do sada najveće pronadjene. Algoritam se bazira na pojmu strategije Redukcije koja je osnovni deo metode Grananja i ogradjivanja. Prikaz rada [195] je dat u Poglavlju 3 u odeljku o Postojećim metodama iz literature.
Algoritam 22 takodje upotrebljava strategiju Redukcije, ali dodatno poboljšane i izmenjene uz dodate funkcije lokalne pretrage i pretrage po okolinama. Prva i treća Redukcija obezbedjuju da se odbacuju čvorovi koji ne zadovoljavaju uslove Teoreme 5.4.1. Druga Redukcija omogućava da se izbegnu ponavljanja pretraga. Peta Redukcija isključuje čvorove iz okoline čvora $N(v_i)$ koji ne zadovoljavaju uslove Teoreme 5.4.1. Četvrta Redukcija je preuzeta iz osnovnog Carraghan-Pardalosovog algoritma i ona testira da li skup U zajedno sa čvorovima koji su već selektovani može da formira kliku veću od najveće sa kojom se do sada već susretalo [51].
MaxClique procedura je glavna procedura (Algoritam 22) i ona za svaki čvor $v_i \in V$ grafa $G = (V, E)$ odredjuje skup $U \subset N(v_i)$ sastavljen od suseda čvora v_i koji su preživeli Redukcije 1, 2 i 3. Sa tako odredjenim skupom kandidata zove se potprocedura Clique. MaxClique iterira po svim čvorovima grafa tako što redom selektuje jedan po jedan čvor, pronalazi okolinu čvora koja predstavlja kandidate koji mogu da čine maksimalnu kliku u kojoj se nalazi dati selektovan čvor. Glavna procedura kao rezultat vraća broj (veličinu) maksimalne klike, tj. maksimum klike. Kriterijum po kojem se vrši izbor sledećeg čvora jeste kriterijum stepena čvora višeg reda. Kada se selektuje takav pogodan čvor smešta se u skup koji je za to predvidjen (u algoritmu opisan sa U). Pomoćna procedura Clique je definisana preko tri argumenta: graf G za koji se traži maksimalna klika, skup čvorova kandidata U, i broj do sada selektovanih čvorova za maksimalnu kliku. U ovoj potproceduri se rekurzivno poziva svaka relevantna klika koja sadrži v_i i na taj način se odredjuje veličina maksimalne klike. Redukcije 4 i 5 smanjuju prostor koji treba da se ispita i na taj način se ubrzava pretraga. Na kraju procesa, vrši se sabiranje svih čvorova maksimalne klike i ispitivanje da li je to najveća kardinalnost.

5.4.4 Brzi heuristički algoritam višeg reda i Metoda promenljivih formulacija višeg reda

Metoda FHHO data u Algoritam 23 [138] je heuristika koja je zasnovana na prethodno opisanom egzaktnom Algoritmu 22, FEHO. Procedura MaxCliqueHeu kao argumente uzima graf G i broj koji predstavlja donju granicu za kardinalnost klike. Kao rezultat procedura vraća broj, tj. kardinalnost klike. Procedura MaxCliqueHeu iterira po svim čvorovima grafa $G = (V, E)$ i za svaki čvor $v_i \in V$ odredjuje se skup U relevantnih čvorova kandidata. Relevantni čvorovi su oni kojima je stepen višeg k-tog reda veći od stepena kardinalnosti klike, dakle selekcija čvorova

je na osnovu stepena višeg reda. Nakon toga se primenjuje Redukcija 1, a dokle god postoje čvorovi u okolini primenjuje se Redukcija 2 i u svakoj okolini čvora ispituju se stepeni čvorova višeg reda. Kada su kandidati preživeli Redukcije 1, 2, i 3 značajni su za skup U koji ih čuva i iz koga se izoluju čvorovi koji učestvuju u maksimalnoj kliki. Nakon treće Redukcije poziva se potprocedura CliqueHeu. Potprocedura CliqueHeu sadrži skup U kao argument i primenjuje dve Redukcije u sebi. Redukcija 4 je primenjena dokle god veličina skupa U nije maksimalna. Zatim se iz skupa U selektuje samo jedan najperspektivniji čvor tako što se zove potprocedura Select, koga priključuje u do sada selektovane čvorove za maksimalnu kliku. To je znatno brže nego da se rekurzivno prolazi kroz sve čvorove kao u FEHO ali je moguće da se preskoči grana pretrage gde se nalazi maksimalna kika pa da se pronadje suboptimalno rešenje. Na kraju ove potprocedure poziva se rekurzivna potprocedura Clique nad kojom se primenjuje Rekurzija 5.
Metoda FSSHO je heuristika koja je zasnovana na promenljivim formulacijama Algoritma 24 [139]. Prostor formulacija može biti definisan na različite načine.
Glavna procedura jeste MaxCliqueHeuMF koja za ulaz ima graf G i broj koji predstavlja donju granicu. Procedura se zasniva na prostoru formulacija kojih ima onoliko koliko je zadato konstantom β. Definisan je prostor formulacija pomoću poznate Motzkin-Straus teoreme s tim što je funkcija ograničenja napisana tako da predstavlja skup formulacija s obzirom da u ovom slučaju ograničenja zavise od parametra β kao u Formulacijama 5.7 i 5.9. I kao u pomenutom radu, fiksirana je funkcija cilja a menjaju se funkcije ograničenja. Akcenat je na promeni politopa na kojima se vrši optimizacija, što je sa jedne strane dovelo do prostora različitih formulacija a sa druge strane dovodi do razvoja matematičkog programiranja u pravcu razmatranja ograničenja na raznim poliedrima. Zato se u okviru procedure defniše parametar β i funkcija prelaza (translacije) rešenja. Dve formulacije se razlikuju po parametru β, odnosno razlikuju se u ograničenju dopustivog skupa rešenja.

Definicija 5.4.2 *Funkcija prelaza jednog rešenja u drugo je maksimum rastojanja dve formulacije. Dve formulacije ϕ_1 i ϕ_2 su na rastojanju k, $d(\phi_1(\beta_1), \phi_2(\beta_2)) = k$ akko $|\beta_2 - \beta_1| = 10^{-k}$, $k \in N$*

U slučaju FSSHO algoritma testiran je skup od 10 formulacija, odnosno 10 promena β parametra i $k = 0, .., 9$. Parametar β menja se u koracima od 0.1 i on ide od 1.1 do 1.9 uključujući i 1.0 kao deseti parametar.
Kako je ukupan broj formulacija 10, a razmatrale su se razlike svaka dva β parametra, broj rastojanja je 45, odnosno $\binom{n}{k}$. Na početku metode postavlja se inicijalno rešenje koje je ulazni parametar algoritma i to je donja granica za veličinu klike. U slučaju da se donja granica ne postavi podrazumevana vrednost je nula. Dalje, dodeljuje se donja granica trenutnoj aproksimaciji veličine maksimuma klike, pa se zatim primenjuje Redukcija 1 koja predstavlja poboljšanu verziju prethodnih objavljenih metoda, tj. ispituje se primena stepena čvorova višeg reda bazirana na Teoremi 5.4.1.
Stepen reda nula iznosi jedan za sve čvorove po definiciji. Stepen reda jedan iznosi koliko je i običan stepen čvora. Stepen reda dva se izračunava na osnovu stepena reda jedan tako što se sumiraju stepeni reda jedan svih elemenata iz okoline nekog čvora, itd.
Procedura vraća broj koji predstavlja donju granicu kardinalnosti klike. MaxCliqueHeuMF koristi pomoćne procedure. Prva pomoćna procedura jeste MaxCliqueHeuMFHelper koja u sebi sadrži argumente grafa G, donje granice i parametra β. Ona radi tako što prolazi po svim čvorovima i ispituje stepene čvorova višeg reda, gde prioritet daje čvorovima sa najvećim stepenom, slično kao i u prethodnim slučajevima.
Skup U se postavlja na početku kao prazan skup i on predstavlja skup u koga se smeštaju čvorovi koji su susedni sa trenutnim čvorom i koji imaju stepene reda k veće od C^k prema uslovima Teoreme 5.4.1, gde je C veličina najveće klike sa kojom se do sada susrelo. U okviru potprocedure poziva se sledeća pomoćna procedura CliqueHeu koja ima argumente graf G, skup U i parametar β. Ona u svakoj iteraciji selektuje po jedan čvor iz U. U okviru pomenute procedure postoje dva kriterijuma za selekciju čvorova, preko parametra β i preko stepena čvora višeg reda. Skup formulacija opisan je parametrom β koji varira od 1 do 2 i u zavisnosti od njega, menja se skup dopustivih rešenja, odnosno kreira se skup prostora rešenja i prostora formulacija. Za $\beta = 1$ dopustivi skup rešenja se nalazi na L_1 sferi, a u opštem slučaju kada β prolazi p dobija se L_p sfera. Time se menja geometrija prostora što omogućava relaksacije u okviru globalne

optimizacije i omogućava primenu različitih numeričkih optimizacionih metoda koje se koriste kako bi se pronašao minimum ili maksimum funkcije cilja u višedimenzionom prostoru.
Funkcija prelaza jeste suštinska u ovom algoritmu s obzirom da ona vrši transformaciju rešenja iz jedne formulacije u drugu i da se u momentu kada se pomoću translacije izvrši pomeranje u sledeći vektor rešenja, pretraga čvora najvišeg reda nastavlja. Podsećamo da zahvaljujući teoremi Motzkin-Straus, čvorovi u grafu se posmatraju kao komponente vektora sa odredjenim vrednostima. Selektovani čvor v iz skupa U se isključuje i uključuje u trenutnu aproksimaciju klike, zatim se traži presek U sa okolinom čvora $N(v)$ i rekurzivno se poziva ponovo procedura CliqueHeu. Maksimalnu kliku čine čvorovi koji su selektovani i isključeni iz U.
Vektor rešenja se računa preko Formule 5.3, Motzkin-Straus.
Pomoćna procedura InitMF izvršava $L_{\max}$ iteracija generalisanog Motzkin-Straus algoritma 23 i tako inicijalizuje vektor x koji se koristi kasnije u proceduri Select. Procedura Select prima jedan argument, skup čvorova kandidata U i odabira jedan element gde na taj način usmerava pretragu u najperspektivnijem pravcu bez da se sistematski ispituje ceo prostor. Iterira se po svim elementima skupa U i odabira se onaj element iz U kome odgovara komponenta koja je najveća u vektoru x sa tim indeksom. U ovoj proceduri se koriste dve pomoćne promenljive p i q. One smeštaju indeks i vrednost najveće komponente u vektoru x respektivno.

Algoritam 22 Brzi egzaktni algoritam višeg reda

```
procedure MaxClique(G = (V, E), lb)
    C ← lb
    for i : 1 to n do
        if d_k(v_i) ≥ C^k, k = 1..k_max then                    ▷ Redukcija 1
            U ← ∅
            for each v_j ∈ N(v_i) do
                if j > i then                                    ▷ Redukcija 2
                    if d_k(v_j) ≥ C^k, k = 1..k_max then         ▷ Redukcija 3
                        U ← U ∪ {v_j}
                    end if
                end if
            end for
            Clique(G, U, 1, C)
        end if
    end for
    return C
end procedure

procedure Clique(G,U,size,C)
    if U = ∅ then
        if size > C then
            C ← size
        end if
        return
    end if
    while U ≠ ∅ do
        if |U| + size ≤ C then                                   ▷ Redukcija 4
            return
        end if
        Izabere se neki element u iz U.
        U = U \ {u}
        N'(u) = {w|w ∈ N(u), d_k(w) ≥ C^k, k = 1..k_max}          ▷ Redukcija 5
        Clique(G, U ∩ N'(u), size + 1,C)
    end while
end procedure
```

Algoritam 23 Brzi heuristički algoritam višeg reda

```
procedure MaxCliqueHeu(G = (V, E), lb)
    C ← lb
    for i : 1 to n do
        if d_k(v_i) ≥ C^k, k = 1..k_max then                    ▷ Redukcija 1
            U ← ∅
            for each v_j ∈ N(v_i) do
                if j > i then                                    ▷ Redukcija 2
                    if d_k(v_j) ≥ C^k, k = 1..k_max then         ▷ Redukcija 3
                        U ← U ∪ {v_j}
                    end if
                end if
            end for
            CliqueHeu(G, U, 1, C)
        end if
    end for
    return C
end procedure

procedure CliqueHeu(G,U,size,C)
    if U = ∅ then
        if size > C then
            C ← size
        end if
        return
    end if
    if |U| + size ≤ C then                                       ▷ Redukcija 4
        return
    end if
    u ← Select(U).
    U ← U \ {u}
    N'(u) = {w|w ∈ N(u), d_k(w) ≥ C^k, k = 1..k_max}             ▷ Redukcija 5
    Clique(G, U ∩ N'(u), size + 1, C)
end procedure
```

Algoritam 24 Metoda promenljivih formulacija višeg reda

procedure MaxCliqueHeuMF($G = (V, E)$, lb, k)
  $\beta = 1.1$
  x^* je globalno rešenje
  $\omega \leftarrow lb$
  for $j : 1$ to k **do** ▷ Dato je k formulacija
    $\beta_j = \beta_{j-1} + 0.1$
    $\beta \leftarrow$ SelectBeta(j)
    $t \leftarrow$ MaxCliqueHeuMFHelper(G, ω, β)
    if $t > \omega$ **then**
      $\omega \leftarrow t$
    end if
  end for
  return ω
end procedure

procedure MaxCliqueHeuMFHelper($G = (V, E)$,lb,β)
  InitMF($G = (V, E), \beta$)
  $C \leftarrow lb$
  for $i : 1$ to n **do**
    if $d_k(v_i) \geq C^k, k = 1..k_{\max}$ **then** ▷ Redukcija 1
      $U \leftarrow \emptyset$
      for each $v_j \in N(v_i)$ **do**
        if $j > i$ **then** ▷ Redukcija 2
          if $d_k(v_j) \geq C^k, k = 1..k_{\max}$ **then** ▷ Redukcija 3
            $U \leftarrow U \cup \{v_j\}$
          end if
        end if
      end for
      CliqueHeu($G, U, 1, \beta, C$)
    end if
  end for
  return C
end procedure

procedure CliqueHeu(G,U,$size$,β,C)
  if $U = \emptyset$ **then**
    if $size > C$ **then**
      $C \leftarrow \frac{1}{1-\frac{1}{2}x^{*T}_{\beta_j} A_g x_{\beta_j}}$
    end if
    $x^*_{\beta_j} \leftarrow \frac{x_{\beta_{j-1}}}{\sum_i x_i^{\beta_j}}$
    return
  end if
  if $|U| + size \leq C$ **then** ▷ Redukcija 4
    return
  end if
  $u \leftarrow$ Select(U).
  $U \leftarrow U \setminus \{u\}$
  $N'(u) = \{w | w \in N(u), d_k(w) \geq C^k, k = 1..k_{\max}\}$ ▷ Redukcija 5
  CliqueHeu($G, U \cap N'(u), size + 1, C$)
end procedure

Algoritam 25 Pomoćne procedure korišćene u algoritmu 24

procedure INITMF(G,β)
  $x = (1, ..., 1)^T$
  for $l : 1$ to $L_{\max}$ **do**
    $x_i = (x_i \frac{(Ax)_i}{x^T A_g x})^{\frac{1}{\beta}}, \quad i = 1, N$
  end for
end procedure

procedure SELECT(U)
  $p \leftarrow 0$
  $q \leftarrow 0$
  for $i : 1$ to $|U|$ **do**
    if $x_{U_i} > p$ **then**
      $q \leftarrow U_i$
      $p \leftarrow x_{U_i}$
    end if
  end for
  return q
end procedure

5.4.5 Rezultati

Algorimi su implementirani u programskom jeziku C++. Testiranje je izvršavano na 64-bit Ubuntu Linux radnoj stanici na Intel i3-4010U procesoru na 1.7GHz. Korišćena je gcc verzija 4.9.1 sa nivoom optimizacije -O3. Treba naglasiti da su testovi izvršavani na mnogo slabijoj platformi u odnosu na publikovane rezultate, ali da su i tako dobijene vrlo dobre i uporedive vrednosti. Algoritmi su uporedjivani sa [195] dok su se autori pomenutog rada uporedjivali sa [51] i [182]. Kod za uporedjivanje sa Fast algoritmima [195] je preuzet sa [58].
Predloženi algoritmi su detaljno ispitani na instancama grafova iz realnog života i DIMACS testnim grafovima.
Grafovi iz realnog života su preuzeti sa University of Florida Sparse Matrix Collection [71]. Tabela 5.7 predstavlja osnovnu tabelu grafova sa opisom broja čvorova, ivica i veličine klike. Kolone prikazuju respektivno ime grafa, broj čvorova, ivica i maksimum klike koji je pronadjen do sada.

Tabela 5.7: Osnovna tabela

Graf	$\|V\|$	$\|E\|$	ω
cond-mat-2003	31,163	120,029	25
email-Enron	36,692	183,831	20
dictionary28	52,652	89,038	26
Fault_639	638,802	13,987,881	18
as-Skitter	1,696,415	11,095,298	67
roadNet-CA	1,971,281	2,766,607	4
kkt_power	2,063,494	6,482,320	11
hamming6-4	64	704	4
johnson8-4-4	70	1,855	14
keller4	171	9,435	11
c-fat200-5	200	8,473	58
brock200_2	200	9,876	12

Tabela 5.8 predstavlja za date grafove rezultate Fast egzaktnog algoritma iz [195] u poredjenju sa FEHO.
Kolone predstavljaju respektivno ime grafa, veličinu maksimuma klike koju je pronašao Fast egzaktni algoritam, vreme koje je bilo potrebno da se algoritam izvrši [195], zatim, veličinu maksimuma klike koju je pronašao FEHO i vreme koje je bilo potrebno da se FEHO algoritam

izvrši. Kao što se vidi iz rezultata, vreme FEHO je bolje u odnosu na vreme koje je predložio Fast. Veličine klike su identične.
Tabela 5.9 predstavlja za date grafove poredjenje rezultata Fast heuristike iz [195] sa FHHO i FSSHO. Kolone predstavljaju respektivno ime grafa, veličinu klike koje je predložila Fast heuristika, vreme koje je bilo potrebno da se Fast algoritam izvrši, veličinu klike koju je dobila FHHO, vreme koje je bilo potrebno da se algoritam izvrši, veličinu klike koju je dobila FSSHO. Poslednja kolona predstavlja vreme koje je bilo potrebno da se FSSHO izvrši. Kao što se vidi u tabeli, u dva primera su pronadjeni maksimumi klike veće kardinalnosti. U većini slučajeva se FHHO izvršava brže u odnosu na Fast algoritam. Rezultati Algoritma 23 se dobijaju na osnovu pronadjene veličine klike i na osnovu vremena za koliko je potrebno pronaći kliku. Kod ove heuristike se ne pretražuje ceo prostor već se na osnovu heuristika bira put koji je najefikasniji za pretragu. Kreće se od jednog čvora pa se u njegovoj okolini bira sledeći čvor koji se ubacuje u kliku po principu stepena čvora višeg reda. U testiranju se najefikasnije pokazao na pomenutim primerima stepen čvora reda dva. Rezultati Algoritma 24 se dobijaju na osnovu sličnog pristupa kao što je gore navedeno. Pravilo na osnovu koga se bira put pretrage se promenilo. Nije uziman čvor stepena višeg reda već su posmatrane komponente vektora $x^{(k)}$ iz Algoritma 24. Birao se čvor koji odgovara komponenti sa najvećom vrednošću $x^{(k)}$ što je k-ta iteracija algoritma GMS prema Jednačini 5.7. U dva primera (as-Skitter, email-Enron) su se dobili bolji rezultati nego u prethodna dva predloga algoritama. Dakle, iako je vremenski najzahtevniji, FSSHO jedini omogućava dobijanje rešenja boljeg kvaliteta u odnosu na state-of-the-art rezultate. Iako kod heuristika ne postoji garancija da će se pronaći najbolje rešenje s obzirom da pretraga nije potpuna, predložena heuristika je omogućila bolji prostor pretrage kako je menjan parametar β. Parametrom β se dobija mogućnost da se menja L_p sfera i da se menja geometrija ograničenja. Kada je $\beta = 1$ ograničenje se svodi na simpleks, dok je za $\beta = 2$ ograničenje sfera. Sve izmedju ove dve vrednosti se svodi na optimizaciju funkcije cilja koja se nalazi na različitim politopima.

Tabela 5.8: Poredjenje egzaktnih metoda

Graf	ωE_{Fast}	$CPUE_{Fast}$	ωE_{FSS}	$CPUE_{FSS}$
cond-mat-2003	25	0.0110	25	**0.0029**
email-Enron	20	0.9980	20	**0.7612**
dictionary28	26	<0.0100	26	**0.0006**
Fault_639	18	20.0300	18	**18.1988**
as-Skitter	67	3838.3600	67	**2541.2800**
roadNet-CA	4	0.4400	4	**0.0290**
kkt_power	11	2.2600	11	**0.4122**
hamming6-4	4	<0.0100	4	**0.0012**
johnson8-4-4	14	0.2300	14	**0.2121**
keller4	11	23.3500	11	**22.6226**
c-fat200-5	58	0.9300	58	**0.5747**
brock200_2	12	1.1000	12	**0.9020**

Tabela 5.10 predstavlja uporedjivanje broja primena Redukcija izmedju [195] i FEHO. Za FEHO je relevantno vreme za koje se vrši pretraga rešenja s obzirom da je dosta solidna brzina pretrage. Slično kao u tehnici grananja i ogradjivanja, bitno je bilo ograničiti prostor pretrage i iskoristiti ograničenja. Tehnika Redukcije je iskorišćena kako bi se ograničio prostor pretrage, odnosno što je veći broj Redukcija, manji je prostor koji je potrebno ispitati, i time predloženi egzaktni algoritam dolazi do rešenja brže.
Kolone predstavljaju respektivno ime grafa, zatim broj primena Redukcija 1, Redukcija 2, Redukcija 3 i Redukcija 5 Fast algoritma. Slede brojevi primena Redukcija 1, 2, 3, 5, kao i 1.1, 3.1, 5.1, kod FEHO. Vrednosti 1.1, 3.1 i 5.1 prikazuju broj Redukcija koje su posledica provere uslova za stepene čvora višeg reda. Ukoliko ove Redukcije postoje može se očekivati da će Algoritam 22 raditi bolje od Fast algoritma. Potvrda ove pretpostavke se vidi na primeru kkt_power gde postoji značajan broj Redukcija 1.1 3.1 i 5.1 i kao posledica ograničenog prostora pretrage FEHO je došao do rešenja za 0.41 sekundu dok je Fast algoritmu trebalo 2.26 sekundi što predstavlja ubrzanje od 5.5 puta. U testiranjima koja su obavljena korišćen je stepen čvora drugog reda. Redukcija je primenjena kada stepen čvora reda k nije bio veći od C^k gde je C kardinalnost

Tabela 5.9: Poredjenje heurističkih metoda

Graf	ωH_{Fast}	$CPUH_{Fast}$	$\omega H1_{FSS}$	$CPUH1_{FSS}$	$\omega H2_{FSS}$	$CPUH2_{FSS}$
cond-mat-2003	25	**<0.0100**	25	0.0110	25	0.1187
email-Enron	18	0.2610	18	0.2134	**19**	**0.6822**
dictionary28	26	**<0.0100**	26	0.0031	26	0.1331
Fault_639	18	5.8000	18	**5.6037**	18	30.5882
as-Skitter	66	27.0800	66	91.3397	**67**	**201.9890**
roadNet-CA	4	0.0800	4	**0.0375**	4	4.4353
kkt_power	11	1.8300	11	**0.7887**	11	8.4457
hamming6-4	4	<0.0100	4	**0.0002**	4	0.0006
johnson8-4-4	14	<0.0100	14	**0.0010**	14	0.0109
keller4	11	<0.0100	11	**0.0052**	11	0.0305
c-fat200-5	58	0.0400	58	**0.0293**	58	0.1253
brock200_2	10	<0.0100	10	**0.0075**	10	0.0214

maksimalne do sada pronadjene klike.

Tabela 5.10: Poredjenje Redukcija

Graf	$P1_{Fast}$	$P2_{Fast}$	$P3_{Fast}$	$P5_{Fast}$	$P1_{FSS}$	$P1.1_{FSS}$	$P2_{FSS}$	$P3_{FSS}$	$P3.1_{FSS}$	$P5_{FSS}$	$P5.1_{FSS}$
cond-mat-2003	29,000	48,000	6.527	17,000	29,407	511	36,171	4,386	449	17,346	251
email-Enron	32,000	155,000	4,060	8,000,000	32,462	111	153,934	3,800	12	8,835,699	306
dictionary28	52,000	4,353	2,114	107	52,139	356	902	262	33	107	3
Faulf_639	36	13,000,000	126	1,116	36	0	13,987,719	126	0	1,116	0
as-Skitter	1,000,000	6,000,000	981,000	737,000,000	-	-	-	-	-	-	-
roadNet-CA	1,000,000	1,000,000	370,000	4,302	1,487,640	454,588	59,994	4,615	4,965	696	532
kkt_power	1,000,000	4,000,000	401,000	2,000,000	1,166,311	33,068	3,778,347	390,985	26,291	1,977,335	23,473
hamming6-4	0	704	0	0	0	0	704	0	0	0	0
johnson8-4-4	0	1,855	0	0	0	0	1,855	0	0	0	0
keller4	0	9,435	0	0	0	0	9,435	0	0	0	0
c-fat200-5	0	8,473	0	0	0	0	8,473	0	0	0	0
brock200_2	0	9,876	0	0	0	0	9,876	0	0	0	0

6

Primene na velikim bazama podataka

U ovom poglavlju predstavljene su odredjene strukture podataka koje su pogodne za detekciju maksimalnih klika. Kasnije su opisane vrste velikih grafova (baza podataka) i primena na analizu društvenih mreža (engl. social network analysis).

6.1 Molekulske strukture

6.1.1 Slaganje

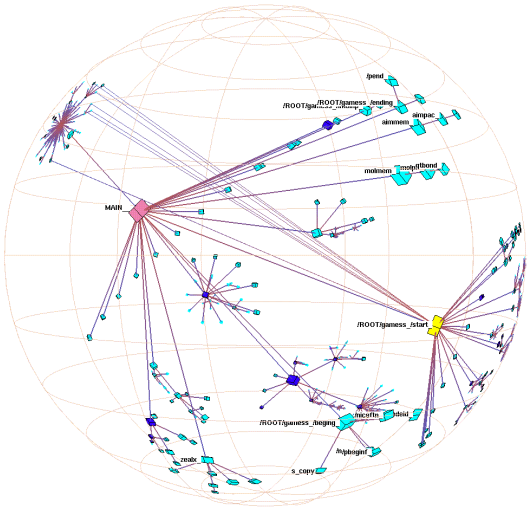
Zajednički podgraf dva grafa G_1 i G_2 se sastoji od podgrafova G'_1 i G'_2 od G_1 i G_2 respektivno, tako da je G'_1 izomorfno sa G'_2. Najveći takav zajednički podgraf je maksimum zajedničkog podgrafa (engl. Maximum common subgraph, MCS). Za dati par grafova, $G_1 = (V_1, E_1)$ i $G_2 = (V_2, E_2)$ njihov pridruženi graf R ima sve moguće parove (v_1, v_2), gde je $v_i \in V_i$, $i = 1, 2$, skup čvorova; dva para čvorova (v_1, v_2) i (v'_1, v'_2) su povezani u R ako su vrednosti ivica od v_1 ka v'_1 u G_1 i od v_2 ka v'_2 u G_2 iste. Za vrednost ivice se najčešće uzima težina ivice. Za par trodimenzionih hemijskih molekula, MCS je definisan kao najveći skup atoma koji imaju odgovarajuće distance izmedju atoma (date su vrednosti tolerancije). Problem nalaženja MCS se može rešiti efikasno koristeći algoritam detekcije klike primenjen na bilo koji graf [89].

6.1.2 Makromolekulske strukture

Problem proteinskih struktura (engl. Protein docking problem) je da se pronadje da li su proteini u interakciji da formiraju stabilan kompleks i ako jesu, kako ga formiraju. Ovaj problem je fundamentalan u svim aspektima biološkog funkcionisanja i razvoj pouzdane tehnike proteinske strukture jeste važan cilj. Jedan od pristupa problemu makromolekulske strukture se sastoji od predstavljanja svakog od proteina preko skupa potencijalnih hidrogenskih lanaca donora i akceptora a koristeći algoritam detekcije klike da se nadje maksimalno komplementaran skup parova donora/akceptora [90].

6.1.3 Mapiranje podataka genoma

S obzirom na razlike u metodama koje se koriste za analizu preklapanja i ispitivanja podataka, integracija ovih podataka je postala važan problem. Preklapanje podataka može da se efikasno konvertuje u ispitivanje elemenata podataka traženjem maksimalnih skupova uzajamno preklopljenih klonova [119]. Svaki skup odredjuje mesto u genomu koje je u vezi sa regionom koji je zajednički kolonovima u skupu; zbog toga se ovi skupovi nazivaju virtuelni ispitivači. Nalaženje virtuelnih ispitivača je jednako nalaženju maksimalnih klika u grafu.



Slika 6.1: Graf poziva

6.1.4 Komparativno modelovanje proteinske strukture

Rapidni porast broja poznatih proteinskih struktura zahteva konstrukciju adekvatnih komparativnih modela. Ovo se može uraditi koristeći pristup detekcije klike. Modelovanje se sastoji od konstrukcije grafa gde su čvorovi koji su u vezi uskladjeni sa ostatkom u sekvencama amino kiselina, a i ivice povezuju parove ostataka čvorova [201]. Optimalna kombinacija klike su klike sa najvećim težinama.

6.2 Veliki skupovi podataka

6.2.1 Primene

Veliki skupovi podataka se javljaju u okviru naučnih, inženjerskih i komercijalnih aplikacija. One uključuju vladine i vojne telekomunikacione medicinske, biotehnološke, astrofizičke, finansijske, ekološke, farmaceutske sisteme, kao i mnogobrojne druge. Neki od širokog opsega problema koji su povezani sa velikim skupovima podataka su skladištenje podataka, kompresija podataka, vizualizacija, pronalaženje informacija, klasterovanje, prepoznavanje paterna, pretraga najbližeg suseda. Sredjivanje ovih problema iziskuje specijalan interdisciplinirani trud u razvoju novih sofisticiranih tehnika. Velike baze podataka nose sa sobom kompleksnu i sadržajnu problematiku a samim tim predstavljaju izazovno polje za rešavanje. U mnogim slučajevima, veliki skupovi podataka se reprezentuju kao veliki grafovi sa odredjenim atributima koji su pridruženi čvorovima i ivicama. Ovi atributi mogu da sadrže specifične informacije koje karakterišu datu informaciju. Analiziranje strukuture takvog grafa je važno da bi se razumela strukturalna karakteristika aplikacije koja se reprezentuje, kao što je poboljšanje pretrage informacija i organizacija memorije. U ovom poglavlju predstavljene su trenutne tendencije u analiziranju velikih grafova. U odeljku o grafovima tržišta predstavljeno je istraživanje koje je u vezi sa grafovima koji su dobijeni na osnovu baza podataka tržišta telekomunikacije.

6.2.2 Modelovanje i optimizacija

Za primere optimizacije grafa modelovanja najčešće se u literaturi koriste baze iz oblasti telekomunikacija. Kao i ranije, $G = (V, E)$ jeste prost neusmeren graf sa skupom čvorova V i skupom ivica E. Takodje, prilikom modelovanja nekog grafa dosta često se koriste pojmovi kao što su distanca i diametar u grafu. Distanca izmedju dva čvora jeste broj ivica na najkraćoj stazi izmedju njih (jednaka je beskonačno za čvorove koji pripadaju različitim komponentama povezanosti). Diametar grafa G se obično definiše kao maksimalna distanca izmedju parova čvorova od G. U slučaju nepovezanog grafa uobičajena definicija diametra rezultuje beskonačnim diametrom,

tako da prateća definicija radi. Diametrom nepovezanog grafa se smatra maksimalna konačna najkraća dužina staze u grafu (što je isto kao i najveći diametar unutar komponenti u grafu).

6.2.3 Primeri velikih grafova

1.Graf poziva (engl. Call graph)
Graf poziva predstavlja graf velike društvene mreže dobijen na osnovu uglavnom velikih baza podataka veličine i do nekoliko terabajta. Podaci su skladišteni iz zapisa na sistemima telekomunikacionih operatera koji uglavnom imaju i po nekoliko miliona korisnika. Ono što se najčešće prati su: broj poziva po korisniku, ukupni minuti razgovora po korisniku i različit broj poziva partnera po korisniku i to sve za odredjeni period vremena. Primer grafa poziva ilustrovan je an slici **??**. Kada baza podataka sadrži sve korisne informacije, pristupa se kreiranju grafa u kom su čvorovi korisnici, veze su skup medjusobnih poziva, a ostale izmerene slučajne veličine su atributi čvorova koji se različito mogu iskoristiti.
Osim što se problem socijalnih mreža posmatra grafovski, mere mu se sve statističke veličine, pa se tako počinje sa odredjivanjem distribucije, testiranjem odstupanja od normalne raspodele, testiranjem značajnosti slučajnih veličina, itd. Kada se dobiju pojedini rezultati moguće je razviti bilo koji algoritam na takvom grafu, uraditi modelovanje i predikciju.
Ovakav graf je ključan zbog toga što govori mnogo o socijalnom ponašanju korisnika, o prostorno-vremenskoj iskorišćenosti mreže i o razumevanju strukture mreža unutar takvog grafa.

2.Internet i Web grafovi
Uloga interneta u modernom svetu nije laka da se tačno odredi i definiše, a posebno da se izračuna i statistički izmeri; suvišno je opisivati da je otkriće globalne mreže uticalo na način interakcije, učenja i komunikacije kao nikad pre. Uporedo sa rastućim značajem, Internet je nastavio da raste kao nadmoćan medij. Visoka dinamika i nepredvidjena struktura svetske globalne mreže privlači sve više pažnje naučnika iz raznih oblasti, uključujući i teoriju grafova. Kao struktura podataka Svetska globalna mreža (engl. WWW) predstavlja usmereni graf gde su čvorovima pridružene strane WWW, a ivicama koje povezuju čvorove su pridruženi hiperlinkovi.

Slično kao i graf poziva, mreža sačinjava multigraf, iako je često tretirana kao neusmeren graf kako bi se pojednostavila analiza. Drugi graf je pridružen fizičkoj mreži Interneta, gde su čvorovi ruteri koji upravljaju paketima podataka ili grupama rutera i domena. Ivice u ovom grafu predstavljaju žicu u fizičkoj mreži. Internet topologija je proizvod Internet projekta mapiranja [55]. Projekat je kreiran iz podataka koji su dobijeni iz nacrta rute iz jednog domena ka skupovima drugih Internet domena. Teorija grafova se primenjuje za web pretragu [33], web majning [166] i druge aktuelne probleme koji se javljaju u vezi sa Internetom.

Algoritmi spoljne memorije

U mnogim slučajevima podaci koji su pridruženi velikim grafovima, su preveliki da bi se smestili u unutrašnju memoriju računara, tako da je potrebno koristiti sporiju spoljnu memoriju (SM). Input/Output komunikacija (I/O) izmedju ovih memorija može rezultovati sporijim izvršavanjem algoritma. Spoljnja memorija algoritma i strukture podataka su dizajnirani sa ciljem da se redukuju troškovi I/O iskorišćenjem lokalnosti. Skorije, algoritmi spoljne memorije se uspešno primenjuju za rešavanje grupa problema koji uključuju grafove, povezane komponente, topološko sortiranje i najkraće puteve.

6.2.4 Problemi prekrivanja

Dat je neusmeren graf $G = (V, E)$ povezan sa ovim problemom. Skup čvorova $V = \{1, 2, ..., n\}$ je u vezi sa skupom traženih tačaka. Skup ivica E se konstruiše. Razmatran je skup $T = \{t_1, t_2, ..., t_n\}$ pločica, svaka je centrirana u traženoj tački. Tako su dva čvora i i j povezana ivicom akko $t_i \cap t_j \neq \emptyset$. Kako bi se prekrile tražene tačke sa minimalnim brojem pločica ili kako bi se minimizovao broj dopustivih lokacija, dovoljno je da se reši problem minimalne particije klike u konstruisanom grafu (ili problem bojenja komplementa grafa).
Problemi prekrivanja su izrasli iz teorije lokacije. Dat je skup traženih tačaka i skup potencijalnih mesta za lociranje sadržaja. Za tražene tačke koje su locirane sa predefinisanom distancom iz sadržaja se kaže da su prekrivene tim sadržajem. Problemi prekrivanja se klasifikuju na dve glavne klase [65]: problemi obaveznog prekrivanja treba da prekriju sve tražene tačke sa minimalnim brojem sadržaja; problemi maksimalnog prekrivanja treba da prekriju maksimalni

broj traženih tačaka sa datim brojem sadržaja. Problemi lociranja su takodje klasifikovani preko prirode skupova traženih tačaka i potencijalnih mesta, a svaki može biti diskretan ili kontinualan. U najviše aplikacija oba skupa su diskretna, ali u nekim slučajevima, barem jedan je neprekidan. Brotcorne i drugi [36] su se bavili aplikacijom problema obaveznog prekrivanja koji se javlja prilikom testa citološkog skriniga raka cerviksa, pri kom je skup traženih tačaka diskretan, a skup potencijalnih mesta je neprekidan. Cilj je maksimizovati efikasnost preko minimizovanja broja uočenih lokacija. Područje pokriveno pomoću ekrana može biti kvadratno ili kružno. Treba uočiti slučaj kvadranog ekrana koji se može kretati u bilo kojem od četiri pravca paralelno sa stranama pravougaonog staklenog slajda. U ovom slučaju, potrebno je pokriti dopustivu površinu slajda preko kvadrata - pločice. Da bi se locirala pločica potrebno je da se specificira pozicija centra slajda na pločici.

6.2.5 Graf tržišta

Iako nije baš očigledno, finansijsko tržište se takodje može predstaviti grafom. Za robno tržište jedna prirodna reprezentacija je bazirana na kros korelaciji fluktuacije robnih cena. Graf tržišta može da se konstruiše na sledeći način: svaka roba se predstavlja čvorom, a dva čvora su povezana ivicom ukoliko je koeficijent korelacije pridruženog robnog para predefinisan pragom θ, $-1 \leq \theta \leq 1$. Bazirano na ovoj informaciji, računa se kros korelacija izmedju svakog para robe koristeći formulu [161]:

$$C_{ij} = \frac{\langle R_i R_j \rangle - \langle R_i \rangle \langle R_j \rangle}{\sqrt{\langle {R_i}^2 - \langle R_i \rangle^2 \rangle \langle {R_j}^2 - \langle R_j \rangle^2 \rangle}}$$

gde je $R_i(t) = \ln \dfrac{P_i(t)}{P_i(t-1)}$ definisan dohodak i za dan t. $P_i(t)$ označava cenu robe i za dan t. Koeficijet korelacije C_{ij} može da varira od -1 do 1. Ova distribucija je skoro simetrična oko aritmetičke sredine, što je približno jednako 0.05 [46]. Glavna ideja konstruisanja tržišnog grafa je sledeća. Neka skup finansijskih instrumenata prezentuje skup čvorova u grafu. Dodaje se neusmerena ivica koja povezuje čvorove i i j ako pridruženi koeficijent korelacije C_{ij} jeste veći od θ. Različite vrednosti θ definišu tržišni graf sa istim skupom čvorova, ali različitim skupom ivica. Lako je videti da broj ivica u tržišnom grafu opada kako prag θ raste. Eksperimenti su pokazali da gustina ivica tržišnog grafa opada eksponencijalno [46].

6.3 Komparacija SAS/STAT i VNS metoda

U ovoj studiji analizirana je mogućnost da se poboljša kvalitet donošenja odluka u okviru jedne telekomunikacione kompanije u Srbiji. Za tu svrhu, uporedjivan je kvalitet SAS/STAT procedura sa algoritmom VNS metode i istraživanje je bazirano na tromesečnim realnim podacima prepaid korisnika sistema. Pokazalo se da je pristup baziran na VNS metodi ostvario bolje rezultate sa manjom greškom, čime su donosioci odluka dobili preciznije smernice i predloge [134].

6.3.1 Uvod

Istraživanje na temu primene VNS algoritma i klika u grafovima je realizovano kako bi se pokazao značaj naprednih tehnologija u poslovnoj analizi i poslovnom odlučivanju, što se praktično ogledalo u formiranju i selekciji podataka, istraživanju podataka, formiranju algoritma, programiranju, statističkoj analizi i konačnom matematičkom modelovanju i predikciji. Zadatak je bio da se detektuju paterni i relevantne informacije za donosioce odluka, s obzirom da se ova vrsta pravila u podacima može koristiti u poslovnoj strategiji i viziji tržišta. Nakon detaljne analize uzorka specifičnih kriterijuma, predloženi su modeli koji minimizuju CPU vreme i učinak, a maksimizuju efekat. Istraživanje je prošireno na nekoliko pravaca, s obzirom da su prepoznati mnogi izazovi matematčkog programiranja i optimizacije. Poslovni modeli su neophodni u svetu vodećih korporacija, u poslovnom prestižu i takmičenju, dakle, neophodni u kreiranju profita, originalnih i jasnih ideja. Dobre prakse pokazuju da bi odluke u velikim kompanijama trebale biti bazirane na primenjenoj matematici, automatskim procesima i procedurama dejta majninga, kako bi se precizno pronašle prave informacije u skladištima baza podataka, tako da se informacije i procene mogu meriti jasno i tačno. Predstavljen je kratak opis analiziranih podataka i modela koji su razvijani sa namerom usvajanja strateškog mehanizma koji bi mogli biti prepoznati i skalirani, tako da su efikasno korišćena najsvežija naučna otkrića i naučni pristupi matematčkog modelovanja u industriji. Telekomunikacione kompanije imaju mnogo mogućnosti da razvijaju alatke za poslovno izveštavanje i da razvijaju modele predikcije za različite oblasti tržišnih zahteva. Zadatak pristupa profilisanju korisnika je organizovan na nekoliko načina:
Statistički - polazi se od deskriptivne statistike (mere centralne tendencije). Mera objašnjava glavne parametre preko Univariate analysis procedure. Procedure su dostupne kroz SAS servere i biblioteke implementirane na SAS programskom jeziku.
Algoritamski - bez statističkih alatki, samo programiranjem metaheurističkih metoda.
Nakon toga, s obzirom na posmatranje pripejd baze podataka korisnika, razvijena je ideja grupisanja i klasifikacije korisnika preko sličnih kriterijuma kako bi se postigla pregledna i lakša segmentacija korisnika koja je uključena u strateški plan specifičnih oblasti kao što su kampanje, rast upotrebe servisa i usluga u kompaniji. Neophodno je pomeriti se iz tačke deskriptivne statistike ka naprednom modelovanju personalizovanih ponuda korisnicima i ka profilisanju svakog pojedinog korisnika kreiranjem lične karte korisnika, uz korišćenje modernih alata za velike strukture podataka, kao što su dejta majning (data mining) i dejta vrengling (data wrangling) ili samo uz korišćenje heurističkih algoritama u prostoru rešenja. Komparativna analiza je pokazala da je VNS metoda ostvarila rezultate sa manjom greškom u odnosu na SAS procedure hijerarhijskog klasterovanja. Predloženi model zahteva manje troškove i vreme izvršavanja, a pri tom proizvodi veći efekat u menadžmentu kompanije.

Opis problema

Formulisanje problema je motivisano kao deo projekta telekomunikacione kompanije zbog kreiranja kampanja, sa ciljem da kampanje treba da se ponude sličnim korisnicima koji se grupišu u nekoliko (maksimalno 10) grupa.
Formalno, neka je S dat skup od n tačaka u prostoru sa koordinatama. U ovom slučaju n predstavlja broj korisnika a kordinate predstavljaju varijable koje ih odredjuju. Neka je $p, p < n, p \in N$. Problem se sastoji od toga da se izabere p (p medijana, centara) od datih n tačaka tako da je suma rastojanja ostalih tačaka do najbližeg centra minimalna.

Publikacije koje su u vezi sa problemom

Profilisanje korisnika i segmentacija korisnika su analizirani kroz mnogo radova tokom vremena. Časopisi koji su publikovali ove teme su naučno orijentisani, ali takodje i industrijski. Neki pristupi koji su izdvajani na tu temu su:
-Vodafone: Customer Segmentation and Customer Profiling for a Mobile Telecommunications

Company (K-means, K-medoid, Fuzzy C-means, The Gustafson-Kessel algorithm, The Gath Geva algorithm) [140]
-Telekom Istanbul, Turkey: Customer churn analysis in telecommunication sector (Based on Genetic algorithm for telecommunication customer subdivision) [107]
-American Telephone and Telegraph Company (ATT): An extended self-organizing map (SOM) network for market segmentation (centroid) [148]
-DIANA, PAM, CLARA, FANNY softverski paketi [210]
Ove studije slučaja su bile razvijane na osnovu poslovnih zahteva zbog poboljšanja tehnologije u odnosu na napredak naučnih dostignuća, pošto su prve tri nabrojane kompanije u okviru organizacione strukture formirale dejta majning departman. Sve ove studije predstavljaju različite algoritme za klasterovanje i predikciju, kao što su K-means, K-medoid, Fuzzy, The Gath-Geva, itd.

Opis podataka

Ulazni podaci su bazirani na dve intervalne numeričke varijable - tromesečni prosek prihoda od dopuna i tromesečni prosek broja dana izmedju dopuna. Pored njih, ulazni podatak je i broj telefona koji je analiziran, i on je dat putem string karakterne promenljive. Podaci su prvo pripremljeni tako sto su standardizovani preko formule $x_{novo} = \frac{x-\mu}{\sigma}$.

Zatim su podaci klasifikovani i analizirane su ekstremne vrednosti. Populacija ima preko 2 miliona korisnika, ali je korišćen slučajni reprezentativni uzorak od 23,111 korisnika (1 % cele populacije) za treniranje modela. Nakon treniranja modela, algoritam je primenjen na celu populaciju uspešno.

Opis korišćenih metoda

Metodologija je planirana na osnovu problematike zahteva. Kako je marketing sektoru bilo potrebno da dobije slične grupe korisnika kako bi postavio različite kampanje, prirodno se zaključilo da je najbolje koristiti klasifikacione (klaster) metode. Klaster analiza je izmedju ostalog, grupisanje skupa objekata na taj način da su objekti u istoj grupi (klasteru) više slični jedni drugima nego objektima u ostalim grupama (klasterima). Izvodjene su najčešće hijerarhijske i particione metode klastera. Pored njih, testirane su metode povezanosti, centroida, distribucije, gustine i graf bazirane metode. Poznata definicija klastera uključuje grupe sa malim rastojanjem izmedju članova klastera, guste površine prostora podataka, intervale i distribucije. U centroid baziranom klasterovanju, klasteri su predstavljeni centralnim vektorom, koji može ali ne mora biti član skupa podataka. Optimizacioni problem je po sebi poznat da je NP težak i poznat pristup je pristup pretrage aproksimativnog rešenja.
Formalno neka je x_i observacija. Neka je D_{KL} bilo koja distanca izmedju centroida klastera. Neka je N_K je broj observacija u klasteru C_K, dok je N_L broj observacija u klasteru C_L. Tada centroid metoda može biti formulisana kao:

$$D_{KL} = \|x_K - x_L\|^2$$
$$D_{JM} = \frac{N_K D_{JK} + N_L D_{JL}}{N_M} - \frac{N_K N_L D_{KL}}{{N_M}^2}$$

gde je $d(x,y) = \|x_K - x_L\|^2$ rastojanje dva centroida.

Opis postavke eksperimenta

Nakon što je definisana metoda, eksperimentalni pristup se odvijao u dva pravca. Prvi se odvijao kroz formiranje klastera u okviru SAS/STAT paketa [49], koji je oficijelno konstituisan kroz softver SAS Enterprise Guide 5.1. Softver SAS EG je zvaničan i dostupan u telekomunikacionoj kompaniji. Usvojena je SAS centroid procedura nakon testiranja mnogih procedura. Pored ove metode, testirane su druge metode koje su definisane u pomenutoj biblioteci, kao što su K-mean, Ward, Average, Single i Density. Programirano je u SAS Base programskom jeziku uz korišćenje procedura statističkih biblioteka za klaster analizu uzorka specifične populacije.
Drugi pravac je metaheuristička VNS metoda [174]. Formulisan je problem kao pretraga p-medijana u kompletnom grafu. Ova formulacija je rešena sa VNS algoritmom u C++/gcc programskom jeziku.

6.3.2 Centroid metoda SAS/STAT biblioteke

SAS/STAT biblioteka sadrži kompletne i predefinisane procedure. Svaka generalizacija klaster analize u SAS paketu treba da se testira s obzirom da postoje brojne metode klasterovanja u različitim industrijama koje se razvijaju u SAS institutu [49]. Definisane su specifične klaster formulacije i specifične klaster funkcije u zavisnosti od hijerarhije klastera. Diversifikacija klaster tehnika se podešava pomoću klasifikacije, klampinga, mašinskog učenja, merenja oblika, podela, sistematizacije, tipologije, prepoznavanja paterna, kvantizacije vektora, itd. U SAS institutu su poboljšali nekoliko tipova disjunktnih klaster metoda i nekoliko tipova hijerarhijskog klastera za poslovne modele. Osim ovih postoje drugi tipovi klastera (fuzzy, overlapping, itd.). SAS je predefinisao procesne procedure za podatke kroz nekoliko formi: matrica koordinata, korelaciona matrica, kovarijansna matrica, L_2 rastojanje izmedju podataka, itd. Najvažnije procedure koje se koriste u SAS klaster biblioteci su: Cluster, Fastclus, Modeclus, Varclus, Tree. U okviru SAS/STAT procedura postoje mogućnosti da se kreiraju drugi tipovi klaster metoda, mada sa aspekta poslovne logike nisu jednostavne, funkcionalne niti efikasne u korišćenju. Neophodno je investirati u pripremu podataka i kvalitet podataka, kao u rezultat evaluacije. Na primer, PROC FASTCLUS se može koristiti za svaki broj klastera. PROC CLUSTER nije dovoljno efikasna za velike brojeve podataka (za baze podataka od terabajtova veličine kao u kompanijama sličnim telekomunikacionim).

6.3.3 VNS metoda

U ovom konkretnom zadatku klasterovanja, primenjena je metoda VNS koja je razvijena za p-medijan problem [174]. Korisnici su tretirani kao dvodimenzione tačke, s obzirom da su definisane dve numeričke promenljive koje definišu klaster. U generalnom slučaju, ako se kreira k varijabli, dobila bi se tačka u k-dimenzionom prostoru. Rastojanje se računa kao euklidsko rastojanje u dvodimenzionom prostoru. Problem se formuliše kao selekcija p tačaka (p centara) iz ovog skupa, pa je suma rastojanja od svake rezidualne tačke do najbližeg centra minimalna. VNS metoda podrazumeva lokalnu pretragu i razmrdavanje, tako da se implementiraju obe faze u p-medijan algoritmu. Lokalna pretraga se sastoji od primene swap(zamene) transformacija, tj. zamenom jednog centra drugom tačkom (ne centrom) dok se poboljšanje ne dostigne. Postoje serije transformacija zamena ne centralnih tačaka u skup dopustivih tačaka, ali jedan centar se mora ukloniti. Ne centralna tačka y traži Best-out tačku (bira se najbolja tačka za uklanjanje). U jednoj iteraciji kreće se kroz skup sa svim tačkama i kreira se funkcija profit-loss. Tačka x je trenutna, njen prvi centar je $c_1[x]$, a drugi $c_2[x]$.

$$profit(x) = d(c_1[x], x) - d(y, x)$$

ako je y bliže nego $c_1[x]$ i 0 inače;

$$loss(x) = \min(d(c_2[x], x), d(y, x)) - d(c_1[x], x)$$

ako je $c_1[x]$ bliže nego y i 0 inače.
Profit se dodaje u jedinstvenu sumu profita, a loss se dodaje gubitku od $c_1[x]$. Target je centar sa minimumom sume gubitka i to je Best-out. Ako je minimum gubitka manji od sume profita, onda se Best-out uklanja iz skupa centara, i u skup centara se dodaje y (bolje rešenje). Nakon promene rešenja, osvežava se prvi i drugi centar za sve tačke. Ako se ne nadje par (Ne-centar, Best-out) kroz lokalnu pretragu, završava se lokalna pretraga. Tokom pretrage koristi se strategija prvog poboljšanja, što znači ako je poboljšanje dostignuto, k se postavlja na inicijalnu vrednost i novo trenutno rešenje se postavlja. Razmrdavanje u k okolina se sastoji od primena k zamena postojećih centara sa nekom drugom tačkom koja nije centar.

6.3.4 Rezultati testiranja

Klasterovanje je primenjeno prvo na 5 klastera, a zatim na 10 klastera, kako je bilo predefinisano zahtevom marketinga. Parametri su rezultirali procedurom klasterovanja na bazi podataka od 23,111 rekorda. Distribucija klastera i parametri se mogu analizirati u donjim tabelama. SAS klaster procedura je ostvarila CPU vremenski parametar od 43 minuta. VNS procedura je

Tabela 6.1: Parametri modela

Parameter	SAS/STAT Centroid 5-cluster	SAS/STAT Centroid 10-cluster	VNS 5-cluster	VNS 10-cluster
Rsquare	0.2160	0.7740	0.5340	**0.7700**
Semipartial Rsquare	0.0014	0.0001	0.0035	0.0001
Cubic Clust Crit	-271	-141	-98	**-56**
Norm Centroid Distance	3.2802	1.2863	3.5889	2.0001

Tabela 6.2: Centri klastera u modelu

Cluster	SAS/STAT Centroid 5-cluster	SAS/STAT Centroid 10-cluster	VNS 5-cluster	VNS 10-cluster
1	0.0021	0.0028	0.0040	0.0057
2	0.0820	0.0795	0.0185	0.0038
3	0.0250	0.0741	0.1599	0.0069
4	-0.4150	0.0698	-0.0007	0.0024
5	-0.0009	-0.0512	-0.2188	0.0412
6	-	-0.0785	-	0.5123
7	-	0.1201	-	-0.4125
8	-	0.1321	-	-0.5514
9	-	0.4152	-	-0.0389
10	-	0.4412	-	0.1214

ostvarila CPU vremenski parametar od 30 minuta i maksimalno razmrdavanje okoline od $k_{max} = p/2$ (p je broj klastera kao i broj medijana).

Kao što se može videti, SAS/STAT procedurom dobija se koeficijent determinacije R-square u iznosu od 21.6%, za metodu od 5 klastera, a za VNS metodu isti parametar iznosi 53%. CCC parametar je deleko od pozitivnih vrednosti u okviru SAS procedure što indukuje da model nije dobro podešen i da može bolje. Kod SAS/STAT procedure (metoda za 10 klastera), R-square je u iznosu od 77.4%, i uporedjujući je sa VNS metodom, poklapaju se. CCC parametar za SAS proceduru je takodje više negativan nego za VNS metodu, što indukuje da model nije dovoljno dobro podešen. CPU vreme izvršavanja je bilo bolje (brže) za VNS metodu nego za SAS metodu. SAS procedure klasterovanja su pokazale da modeli dobijeni preko njih moraju biti poboljšani i ispravljeni, s obzirom da model predlaže korektne rezultate samo za više od 30 klastera. To je zaista veliki broj klastera s obzirom da je marketing sektor jasno naglasio granicu do koje se može ići. Kad ne bi toga bilo, poslovni sektor bi trebalo da smisli više od 30 kampanja, što bi bilo skupo. Analitičari podataka treba da naprave intervencije nad podacima i da kreiraju veštački ulaze kako bi dobili dobre rezultate sa ovim paketom koji je definisan u biblioteci. Kreiran je jedan način veštačkog modela tako što je klasterovan prvi klaster rekurzivno, s obzirom da je ovaj klaster najproblematičniji. Na ovaj način model je dao zadovoljavajući rezultat, iako je prividan i kodiran pristrasno.

6.3.5 Evaluacija klasterovanja

Evaluacija klasterovanja je definisana preko različitih metodologija. Prvo, uporedjivanje SAS/STAT i VNS procedura se započinje posmatranjem brojem klastera. To je pokazalo da SAS/STAT klasterovanje nije dalo korektan broj klastera, odnosno upotrebljiv broj klastera, za razliku od VNS koji ne zahteva veliki broj klastera već grupisanje uradi približno korektno kako za male, tako i za velike brojeve klastera. Pored toga, raspodela u okviru grupa je takodje vrlo problematična za SAS/STAT proceduru u odnosu na VNS proceduru. Analiza je primenjena kroz frekvencije grupa i kroz distribuciju grupa. Nakon toga, grafički su predstavljeni klasteri i pokazalo se da

Tabela 6.3: Distribucija korisnika u 5-klaster proceduri

Cluster	SAS/STAT No. Customer	VNS No. Customer
1	23054	**4944**
2	46	**833**
3	9	**1879**
4	1	**10763**
5	1	**4692**

Tabela 6.4: Distribucija korisnika u 10-klaster proceduri

Cluster	SAS/STAT No. Customer	VNS No. Customer
1	21678	**3862**
2	833	**2614**
3	543	**2405**
4	37	**227**
5	4	**1107**
6	9	**665**
7	3	**3572**
8	2	**833**
9	1	**1605**
10	1	**6221**

za istu SAS/STAT proceduru, oblik klastera ne odgovara realnim grupama podataka, dok za VNS proceduru oblik svih grupa jeste vizuelno razumljivo klasifikovan. Ova analiza se bazira na vizualizaciji. Zaključak iz osnovne evaluacije je da korektnost podele jeste lokalno više optimizovano za VNS nego za SAS/STAT proceduru. Mnogo naprednija evaluacija se izvela kroz analiziranje da li su podaci grupisani u smisleni poredak i to u okviru različitih metoda validacije, kao što su: Partition coefficient (PC), Classification entropy (CE), Partition index (PI), Stability of crossing matrix between segments, itd. Predstavljen je ovde Davies Bouldin indeks i rezultati:

$$DB = 1/n \sum_{i=1}^{n} \max \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

gde je c_i centroid $i$tog klastera, n je broj klastera, σ_i prosečna distanca svih elemenata u klasteru. Dobijen je manji indeks za VNS (za 5 i 10 verzija klastera) nego što je dobijeno za SAS/STAT proceduru, što definiše da je DB indeks za stabilnost klastera mnogo precizniji za VNS metodu. Optimalno rešenje klasterovanja ima najmanju vrednost DB indeksa.

Tabela 6.5: Evaluacija indeksa u modelu

Evaluation indexes	SAS/STAT Centroid 5-cluster	SAS/STAT Centroid 10-cluster	VNS 5-cluster	VNS 10-cluster
Davies Bouldin criterion	5.267	4.874	1.741	**0.785**

6.3.6 Zaključak

Svrha optimizacije i modelovanja klaster metodom je u pametnom algoritmu, s obzirom da su rekordi (tačke, podaci) kodirani na osnovu automatske pretrage u optimalnom vremenu i uz korišćenje optimalne memorije, uz definiciju da su tačke unutar grupe homogene, a da su tačke izmedju grupa heterogene. Ukoliko bi bili potrebni fuzzy klasteri kako bi se mogle pozicionirati specifične tačke sa nekim verovatnoćama u nekoliko različitih grupa, predlaže se model koji može

nezavisno da skorira verovatnoću repozicioniranja u tačkama. Dobro definisan algoritam minimizuje pristrasnost ili statističko ocenjivanje unapred. Statističko ocenjivanje je vrsta devijacije s obzirom da se računa sistematična razlika od konstante populacije. Ovo proizvodi da je model statistički i analitički dobar ako postoji nizak kapacitet unapred stvorenih definisanih parametara i takodje ako je model razvijen bez poboljšavanja rezultata i popravljanja kvaliteta podataka. Dobar model podrazumeva mapiranje izolovanih tačaka koje su značajne i nezavisne, tako da se rezultat realizuje koristeći pravilne informacije iz podataka i iz značajnih kriterijuma istih, bez mnogo spoljnih (ručnih) intervencija analitičara. Uporedo, zvanično prihvaćeni parametri, kao što je koeficijent determinacije, odredjuju stabilnost modela i analitičar procenjuje kvalitet uporedjujući koeficijente. Za industriju je najvažnije da je rezultat upotrebljiv, podesan, fleksibilan i logičan. U ovom slučaju frekvencija i distribucije SAS/STAT klastera nisu logične i korisne, ali jesu na osnovu procedure VNS. Glavni doprinos istraživanju je i taj što nije poznato da li je ijedna telekomunikaciona kompanija na svetu koristila metaheurističke metode kao što je VNS kako bi rešila zadatak klasterovanja. Primena p-medijana VNS algoritma je novi pristup rešavanju klastera i profilisanju korisnika u segmentaciji.

7

Zaključak i budući rad

Maksimalna klika je jedan od prvih problema za kog je dokazano da je NP-težak, a čak su i njegove aproksimacije sa konstantnim faktorima NP-teške, zbog čega su se istraživači koncentrisali na dizajniranje efikasnih heuristika koje dovode do optimalnih ili rešenja bliskih optimalnim, tj. zadovoljavajućeg kvaliteta.

Rani radovi iz ove oblasti datiraju iz sredine sedamdesetih godina kada su Garey i Johnson [91] dokazali da ako problem maksimuma klike zahteva polinomijalno vremensku aproksimaciju algoritma (aproksimativna za konstantan faktor), onda postoji polinomijalno vremenska aproksimativna shema (aproksimativno je za bilo koji proizvoljno mali faktor). Ukoliko problem maksimuma klike pripada klasi APX, onda pripada i PTAS klasi. Johnson [145] je pokazao 1988. da algoritam polinomijalnog kašnjenja za enumeraciju maksimalnih klika u obrnutom leksikografskom poretku (ukoliko je $P \neq NP$) ne postoji.
Na temu aproksimativnih algoritama problema maksimuma klike koji se rešavaju u polinomijalnom vremenu ima dosta istraživanja u literaturi. S tim u vezi, kompleksnost nije u prvom planu, iako je neophodno ispuniti i vremensku efikasnost u oblasti optimizacije. Fokus je na istraživanju praktično preciznog algoritma, ali i teorijski dovoljno zanimljivog algoritma u oblasti globalne optimizacije. U ovoj disertaciji predstavljeni su novi efikasni pristupi za rešavanje maksimuma klike u grafu i maksimalnih klika u grafu, i s tim u vezi za rešavanje ekvivalentnih problema, stabilnog skupa i minimalnog čvornog pokrivača. Cilj istraživanja je bio da se reše odredjeni problemi klike: težinska klika na čvorovima, particionisanje klike i netežinska klika definisana preko kontinualnih funkcija. Svaki od problema rešavan je preko različitih heuristika, najčešće preko Metode promenljivih okolina. Pored promene okolina primenjene su i promene formulacija. U ranijim radovima je pokazano da je VNS dosta efikasna metoda u rešavanju NP-teških problema. FSS metoda je izuzetno izazovna za rešavanje kontinualnih nekonveksnih problema, sa stanovišta optimizacije vremena. Osnovno pitanje koje se postavljalo jeste na kakav problem koju varijantu (ili koje varijante) Metode promenljivih okolina razvijati. Na početku svih problema istraživanja pokazalo se da je verovatno najbolje razviti klasičnu (osnovnu) varijantu koja se sastoji od serije razmrdavanja i lokalnih pretraživanja.
Postupak koji je zasnovan na VNS algoritmu za težinsku kliku (težine su dodeljene čvorovima grafa) pokazuje slične karakteristike kao postupak zasnovan na faznom lokalnom pretraživanju. Za odredjene grupe grafova koji su veće gustine, VNS pokazuje bolje rezultate, tako da su upravo tom metodom dobijena najbolja poznata rešenja. Najbolji rezultati su postignuti zahvaljujući primeni kombinacije simplicijalnih čvorova grafa (lokalno pretraživanje koje se koristi u okviru spusta kroz promenljive okoline), kao i primeni podalgoritma penala i podalgoritma stepena, uz korišćenje funkcije prioriteta koja meri značajnost zadržavanja svakog čvora u trenutnoj aproksimaciji maksimalne klike.
Popravljene su karakteristike lokalne pretrage i razmrdavanja. Okoline u lokalnoj pretrazi i razmrdavanju se razlikuju po složenosti pretraživanja i u toku spusta pretražuju se okoline koje su jednostavnije za pretragu, i jedino kada u nekoj okolini nema boljeg rešenja traži se bolje u nekoj drugoj okolini.
Što se tiče problema particije klike uz klasičnu VNS, testirane su i kombinacije GVNS i SVNS, kao i medjusobne interakcije algoritama. Adaptivna metoda promenljivih okolina razvijana

je radi poboljšanja rezultata koji su dobijeni primenom osnovne Metode promenljivih okolina. Naravno, prethodno su provereni svi delovi Metode promenljivih okolina: lokalna pretraga i razmrdavanje. Lokalna pretraga morala je biti što efikasnija, jer bi postupak rešavanja trajao suviše dugo. Razmrdavanje je moralo obezedjivati prelazak u razne delove okoline jer samo na taj način može se sistematično pretražiti prostor rešenja. Kod te metode trebalo je promeniti kriterijum za prelazak iz jednog rešenja u drugo, tako da se dozvoli prelazak i u lošije rešenje.
Pretraga kroz prostor formulacija ili FSS je metoda koja podrazumeva više različitih formulacija problema koji se razmatra. Svaka od njih ima svoju lokalnu pretragu pri čemu se vodi računa o poboljšanju funkcije cilja maksimalne klike. Razmatran je problem nalaženja MC kompletnog podgrafa zadatog grafa uz pomoć različitih formulacija.
U ovom radu su predstavljeni novi rezultati za problem maksimuma klike primenom egzaktnog i heurističkih algoritama. Funkcija cilja je kardinalnost klike, uz ograničenje da su svi čvorovi medjusobno povezani. Postupak koji je prikazivao egzaktan i heurističke algoritme za rešavanje problema maksimuma klike u opštem slučaju je bio zasnovan prvo na primeni stepena čvora višeg reda, a zatim na primeni više formulacija. Formulacije koje smo mi koristili su diskretnog i neprekidnog tipa definisane na osnovu kvadratne funkcije na regularnim politopima, a rezultati su postignuti zahvaljujući FEHO, FHHO i FSSHO algoritmima koji su prilagodjeni tim formulacijama. Algoritmi su predstavljeni u poglavlju o heuristikama koje su primenjene. Tu je prikazana nova ideja kombinacije stepena čvora višeg reda i formulacija, odnosno pretraga rešenja na osnovu promene parametara kojim se menjaju formulacije. FEHO je takodje u sebi sadržao rekurziju i pomoćne funkcije za selekciju čvorova, uz strategije redukcije kojima su se eliminisale čitave grane i delovi grafa. Na osnovu dobrih rezultata u egzaktnom slučaju, kreirana je i heuristika koja je takodje zasnovana na rekurzivnim pozivima selekcije. Selekcija se izvršavala kako na osnovu stepena čvora višeg reda tako i na osnovu skupa formulacija u kojima je variran parametar β i koji je promenom menjan tako da se i cela funkcija cilja menjala. Time je stvoren efekat više formulacija. Zapravo, rekurzivni pozivi funkcije *Select* su ispitivali da li je bolji kriterijum za selekciju stepen čvora višeg reda ili promena funkcije cilja. Kako su rezultati u skoro svim testiranim primerima identični sa blagom prednošću ka promeni skupa formulacija, tako je zaključeno da je FSS metod pogodan za ovako težak problem.
Razmatrano je na osnovu prvih rezultata kako da bude implementirana varijanta promene formulacija. Ostvareni su solidni rezultati na primerima malih dimenzija za FSS metodu. Kod FSS je potrebno menjati kriterijum za prelazak iz jednog rešenja u drugo i definisati prostor na kom su dopustiva rešenja. Što je veće odstupanje od ograničenja veća treba da bude funkcija penala. Istraživano je koliko treba smanjiti σ, standardnu devijaciju koja odredjuje slučajan vektor sa gausovom raspodelom i dijagonalnom kovarijansnom matricom. Posmatrana je optimizacija na raznim poliedrima i testiran simpleks i sfera čija je norma L_p. Pokazalo se da hiperravan dobro balansira procenat poboljšanja, pa postoji 50% šansa da se desi poboljšanje. Kasnije se od hiperravni prešlo na L_p sferu, odnosno na sferu u L_p normi i pokazala uspešnost rezultata za promenu β u okviru konveksnog omotača k-dimenzionog politopa. Obavljeni su brojni eksperimenti na više kategorija grafova i uporedjene performanse ovog algoritma sa drugim autorima. Na osnovu priloženih rezultata FEHO algoritam koji je predložen je uporediv po svim instancama. Za pojedine grafove FEHO algoritam je brži, na primer kod kkt_power instance on je 5 redova veličine brži, čemu su doprinele Redukcije usled poboljšanog kriterijuma za izbor čvora. Na osnovu priloženih rezultata FHHO i FSSHO algoritmi su uporedivi sa publikovanim rezultatima. FHHO postiže iste rezultate za kraće vreme osim u jednoj instanci a FSSHO postiže na dve instance bolje rezultate nego publikovani radovi, dok se u ostalim instancama poklapa sa publikovanim. U primeru as-Skitter je dostignut rezultat koji dostiže i FEHO, dakle tačno rešenje. Redukcije koje su postignute u FSSHO su dosta veće nego u ostalim algoritmima koji su testirani, dakle i publikovanim i predloženim, tako da se zaključuje da je strategija Redukcije zadovoljila značajnost FSSHO. Na primer, instanca roadNet-CA je dostigla značajnu Redukciju prostora pretrage po formulacijama. FSSHO je prikaz prave pretrage po prostoru formulacija tako da se smatra da ovde postoji mesto za dalje istraživanje i poboljšavanje. Time se dokazalo da je pretraga po prostoru formulacija korisna heuristika u slučaju predstavljanja problema kombinatorne optimizacije preko kontinualne nekonveksne funkcije. Nekonveksni kontinualni problemi globalne optimizacije su NP teški problemi i važno je doprineti novim rešavanjima ovakvih funkcija. Kombinovanje pretrage prostora formulacija je dalo brže, efikasnije i tačnije rezultate.

Takodje, jedna od hipoteza značajna za budući rad je kako analizirati i testirati funkcije ograničenja koje u ovom i sličnim slučajevima razmatraju geometrijski pristup i uvode polje poliedarske kombinatorne optimizacije što je jedna od interdisciplinarnih oblasti matematičkog programiranja. Problem maksimalne klike je interesantan zbog svoje direktne povezanosti sa teorijom grafova na osnovu koje se formiraju različite tehnologije - dejta majning, internet i socijalne mreže, prepoznavanje test paterna, izračunavanje delimičnih Bulovih funkcija, teorija kodiranja, itd. Problem maksimalne klike ima mnogo ekvivalentnih formulacija, kao što su celobrojno programiranje i neprekidno nekonveksno programiranje [192]. Predstavljeni su neki postojeći pristupi i dokazi za odredjene formulacije. Počelo se sa celobrojnim formulacijama, zatim su predstavljene mešovite celobrojne formulacije i razmatrane neprekidne formulacije problema maksimalne klike gde su dati i dokazi za neke od njih. Formulacije su generalizovane na sve kompatibilne probleme na grafovima - problem maksimalnog stabilnog skupa i problem dominantnog skupa. Mnogi rezultati koji su prezentovani se mogu pronaći u radu autora Abello-a [5]. Istražujući različite d.c. (difference-convex) formulacije nekonveksnih problema došlo se do ideje za novi pristup u rešavanju ovih zadataka. Problem maksimalne klike je ovde zapisan preko diskretne formulacija ali i preko kontinualne kvadratne nekonveksne funkcije. Kao osnovni i veoma primenljivi u industriji, bili su interesantni autorima koji su radili veliki broj istraživanja čiji je cilj bio pronalaženje efikasnih algoritama za nalaženje približnog (ali i tačnog) rešenja tih problema.
Pokazano je ukratko u poglavlju o maksimalnoj kliki koji različiti algoritmi postoje za njihovo odredjivanje: enumerativni algoritmi, Konstruktuvne heuristike, Genetski algoritmi, Simulirano kaljenje, GRASP, Tabu pretraga, Fazno lokalno pretraživanje, Hibridna promena okolina u tabu pretrazi, itd. Rezultati su pokazali da je VNS u vrhu najefikasnijih algoritama i to po rezultatima koji su "state-of-the-art" na kolekciji zvaničnih test primera i po CPU vremenu koje je potrebno za izvršavanje algoritma. Nisu data poredjenja sa svim metodima ovde, s obzirom da su noviji radovi na tu temu prevazišli mnoge pomenute algoritme i ne prikazuju poredjenje sa tim metodama. Karakteristike tih prvobitnih algoritama su primetno slabije, pa su i rezultati slabiji u odnosu na novije.
Nakon teorijskog pristupa koji je testiran na DIMACS grafovima, predstavljeno je poglavlje o primenama, gde su prikazane praktične primene ovog pristupa u industriji. Diskutovano je o primeni ove i sličnih disertacija na širok spektar problema u industriji, s obzirom da su mnogi problemi u industriji NP-teški i zahtevaju dejta majning alate koji se oslanjaju na algoritamske i metaheurističke pristupe. Ovo je refleksija trenutne tendencije u odnosu na interdisciplinarno istraživanje u oblasti optimizacije i nauke uopšte. Predstavljene su novije tehnike koje postoje u literaturi i koje su do sada najbolje a koriste se za analiziranje strukture telekomunikacionih podataka, finansijskih podataka, internet podataka, i uopšte glomaznih struktura do kojih se došlo u analizi. Strukture podataka su mreže koje predstavljaju tržište bazirano na kros-korelacijama izmedju numeričkih varijabli i ocenjivanju fluktuacija izmedju njih. Cilj primene metaheuristika je ostvarivanje korisnih alata za klasifikaciju instrumenata i kreiranje softvera na osnovu instrumenata merenja. Istraživane su stabilnost i dinamičnost sistema u odnosu na primenu raznih algoritama na tržišnim grafovima i u odnosu na vremenske serije u podacima. Dejta majning je izazovan naučni problem i interdisciplinarni segment računarstva u kome se otkrivaju obrasci u velikim bazama podataka i koji vodi do efikasnog pretraživanja podataka [111]. Preciznije, dejta majning pruža mogućnost izolovanja informacija iz skupa podataka, nakon čega se informacije pretvaraju u razumljivu strukturu za dalju upotrebu. Cilj je bio da se istraživanje primeni na analitiku predvidjanja i predikcionu analizu grupa i klastera velikih baza podataka. Postoji mnogo problema velikih dimenzija u oblasti internet transmisije koji bi trebali biti rešeni. Tačna rešenja i procene za veličinu najvećeg korigovanja greške kodiranja veće dužine mogu biti izračunata dizajniranjem i primenom efikasnijeg, paralelnog algoritma koji rešava optimalno problem maksimalne klike. Jedna zanimljiva primena je takodje rešavanje ad hoc wireless mreže putem rešavanja problema minimalnog povezanog dominantnog skupa u unit-disk grafovima. Pored odličnog izvršavanja predloženog pristupa, jedan od nedostataka je da je dopustivo rešenje konstruisano samo kad algoritam prekida rad. Taj nedostatak se može prevazići razvijanjem novih algoritama koji bi mogli da postave dopustivo rešenje u bilo kojoj fazi njihovog izvršavanja. Drugi bitan problem je istražiti postavljanje povezanog dominantnog skupa u mobilnom okruženju. U nekim od budućih radova moglo bi da se proširi istraživanje na primene u biomedicinskom inženjeringu i istraže razlike u ponašanju podataka u različitim pomenutim sektorima: telekomunikacionom, finansijskom i biomedicinskom.

Budući rad u teorijskom pravcu

Planirano je istraživanje u nekoliko smerova, u zavisnosti od ponašanja algoritma. Jedan smer je povezivanje VNS algoritma sa principom d.c. na sličan način kako ga je definisao Strekalovski. To znači da će biti primenjena na VNS algoritam lokalna pretraga data *Support* funkcijom kako je opisao autor, a suština takve lokalne pretrage jeste da se smanjuje početni skup čvorova i da se povećava skup koji čini kliku. Okoline tog rešenja, kako je definisano VNS algoritmom su skupovi skupova, a njihov broj je potrebno minimizovati, kako bi pretraga u skupu okolina postala jednostavnija [207].

Drugi smer je sličan principu Butenka, odnosno polazeći od Motzkin-Straus teoreme i neprekidne postavke problema traži se način kako da se diskretizuje skup na kome se vrši pretraga rešenja čime se dobija skup diskretnih skupova rešenja i samim tim, skup diskretnih formulacija. Ovaj pristup je interesantan kako zbog teorijske postavke Metode promenljivih formulacija, tako i zbog primene nekonveksne analize u teoriju globalne optimizacije. Naime, naučno je izazovno odrediti funkciju translacije rešenja iz jedne formulacije u drugu. Funkcija translacije je obično nelinearna i potrebno je testirati razne odnose kako bi se diskretan skup konvertovao u kontinualan R^n [48].

Treći smer je interesantan u smislu poboljšanja trenutno efikasnog heurističkog FSS algoritma koji je prikazan u prethodnom opisu. Poboljšanje se zasniva na uvodjenju VND strategije, odnosno, Metode najstrmijeg spusta u okviru Metode pretrage formulacija uz korišćenje čvora stepena višeg reda. Potrebno je izolovati okoline rešenja kako bi se funkcija cilja što više ispravila i kako bi bilo moguće implementirati VND algoritam.

Nadalje, metodologija za analiziranje strukture tržišnog grafa može biti proširena na pretragu klastera u odnosu na vremensku seriju koja bi se računala, a kao ulazna baza podataka bi se mogle koristiti varijable merenja jednog obeležja u dva različita vremena. Dakle, radi se o ponovljenom merenju nad istim skupom podataka i nakon toga, moguće je izračunati matricu kovarijanse koja prikazuje varijansu elemenata kao slučajni vektor. S obzirom da je matrica kovarijanse pozitivno semi-definitna problem rešavanja vremenskih serija se svodi na kontinualnu konveksnu optimizaciju. Projekat bi izmedju ostalog mogao da prezentuje rezultat koji čvorovi će biti profitabilni tokom odredjenog vremenskog perioda, da odredi tip čvorova, kao i specifične informacije (karakteristike) o čvorovima. Sve pomenute ideje se oslanjaju na matematičke formulacije problema klike pri čemu se vodi računa o donjim i gornjim granicama kod klike, o transformaciji kontinualnih funkcija prilikom izračunavanja vrednosti funkcije cilja i s tim u vezi o transformaciji Motzkin-Straus teoreme, koja je zaista veliko otkriće i spona izmedju sveta diskretnih i kontinualnih struktura.

Prilagodjavanje teoreme Motzkin-Straus na razne politope koji predstavljaju prostor rešenja za klike je izazov sam za sebe i moguće je da istraživanje ode u pravcu ispitivanja promena u teoremi prilikom promena geometrijskih struktura. Time se stiču uslovi za odgovore na mnoga pitanja iz oblasti globalne optimizacije, stiču se novi uvidi u veze izmedju kombinatornih i kontinualnih problema, geometrija i teorija poliedara zauzimaju značajna mesta kad je reč o matematičkom programiranju i to ne samo u oblasti linearnog programiranja već su naznake da kombinatorna poliedarska optimizacija može da rešava i kvadratne nekonveksne funkcije.

Ovim zaključujemo da optimizacija kontinualnog nekonveksnog problema zauzima važno mesto ne samo u okviru teorijskog računarstva, programiranja i algoritama, već i u okviru analize, teorije funkcija i geometrije. Očekuje se da će odredjeni radovi u budućnosti u oblasti optimizacije biti zasnovani na istraživanju površi i politopa izmedju ostalog.

Literatura

[1] E. Aarts, J.K. Lenstra *Local search in Combinatorial Optimization*, J. Wiley and Sons, Chichester, UK, 1997.

[2] E. Aarts, J. Korst *Simulated Annealing and Boltzmann Machines*, J. Wiley and Sons, Chichester, UK, 1989.

[3] J. Abello, A. Buchsbaum, J. Westbrook *A functional approach to external graph algorithms*, In Proceedings of the European Symposium on Algorithms, Lecture Notes in Computer Science, Springer- Verlag, Berlin, 332–343, 1998.

[4] J. Abello, J.S. Vitter *External Memory Algorithms*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, RI, 247-277, 1999.

[5] J. Abello, P.M. Pardalos, M.G.C. Resende *On maximum clique problems in very large graphs*, External Memory Algorithms, 50 of DIMACS Series in Discrete Mathematics and Theoretical Computer Scienc, American Mathematical Society, Providence, RI, 119–130, 1999.

[6] L. Adamic *The small world Web*, In Proceedings of ECDL99, 1696 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 443–452, 1999.

[7] L. Adamic, B. Huberman *Power-law distribution of the World Wide Web*, Science, 287, 2115-2116, 2000.

[8] W. Aiello, F. Chung, L. Lu *A random graph model for power law graphs*, Experimental Math., 10, 53–66, 2001.

[9] R. Albert, H. Jeong, A.L. Barabasi *Diameter of the World-Wide Web*, Nature, 401, 130–131, 1999.

[10] L. Amaral, A. Scala, M. Barthelemy, H. Stanley *Classes of small-world networks*, Proc. of National Academy of Sciences USA, 97, 11149-11152, 2000.

[11] A.T. Amin, S.L. Hakimi *Upper bounds on the order of a clique of a graph*, SIAM J. Appl. Math., 22, 569-573, 1972.

[12] L.T.H. An, P.D. Tao *The DC(difference of convex functions* Programming and DCA revisited with DC models of real world nonconvex optimization problems, Annals of Operations Research, 133, 23-46, 2005.

[13] S. Arora, S. Safra *Probabilistic checking of proofs: A new characterization of NP*, Proc. 33. Ann. Symp. Found. Computer Sci., 2-13, 1992.

[14] M. Avriel *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[15] E. Balas, C.S. Yu *Finding a maximum clique in an arbitrary graph*, SIAM Journal of Computing, 1054–1068, 1986.

[16] E. Balas, V. Chvatal, J. Nesetril *On the maximum weight clique problem*, Math. Oper. Res., 12, 522-535, 1987.

[17] A.L. Barabasi, R. Albert *Emergence of scaling in random networks*, Science, 509–512, 1999.

[18] R. Battiti, M. Protasi*Reactive local search for the maximum clique problem*, Algorithmica, 29, 610-637, 2001.

[19] M.S. Bazaraa, C.M. Shetty *Nonlinear Programming*, J. Wiley and Sons, USA, 1979.

[20] A.R. Bednarek, O.E. Taulbee *On maximal chains*, Roum. Math. Pres et Appl., 23–25, 1966.

[21] M. Bellare, M. Sudan *Improved nonapproximability results*, Proc. 26. Ann. ACM Symp., Theory of Comput., 184-193, 1994.

[22] C. Berge, V. Chvatal (Eds.) *Topics on Perfect Graphs*, Ann. Discr. Math., 21, 57-61, 1984.

[23] P. Berman, G. Schnitger *On the complexity of approximating the independent set problem*, Inform. and Comput., 96, 77-94, 1992.

[24] K.P. Bogart *Combinatorics Through Guided Discovery*, National Science Foundation Grant, DUE0087466, 2004.

[25] B. Bollobas, P. Erdos *Cliques in random graphs*, Math. Proc. Camb. Phil. Soc., 80, 419–427, 1976.

[26] B. Bollobas *Random Graphs*, Academic Press, New York, 1985.

[27] I.M. Bomze, M. Pelillo, R. Giacomini *Evolutionary approach to the maximum clique problem: empirical evidence on a larger scale*, Developments of Global Optimization, Kluwer Academic Publishers, Dordrecht, 95–108, 1997.

[28] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo *The maximum clique problem*, Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1-74, 1999.

[29] R.E. Bonner *On some clustering techniques*, IBM J. Res. Develop., 22–32, 1964.

[30] R. Boppana, M. Halldórsson *Approximating maximum independent sets by excluding subgraphs*, Bit, 32, 180-196, 1992.

[31] J. Brimberg, D. Urosevic, N. Mladenovic *Variable neighborhood search for the vertex weighted k-cardinality tree*, European J. of Operational Research, 171, 74-84, 2006.

[32] J. Brimberg, S. Janićijević, N. Mladenović, D. Urošević *Solving the Clique Partitioning Problem as a Maximally Diverse Grouping Problem*, Optimization Letters, 1-13, 2015.

[33] S. Brin, L. Page *The anatomy of a large scale hypertextual Web search engine*, In: Seventh International World-Wide Web Conference (WWW 1998), Brisbane, Australia, April 14-18, 1998.

[34] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener *Graph structure in the web. WWW9*, Computer Networks, 33, 309–320, 2000.

[35] C. Bron, J. Kerbosch *Algorithm 457: Finding all cliques on an undirected graph*, Communications of ACM, 575–577, 1973.

[36] L. Brotcorne, G. Laporte, F. Semet *Fast heuristic for large scale covering location problems*, Computers and Operations Research, 29, 651–665, 2002.

[37] P. Brucker *Scheduling Algorithms*, Universitat Osnabruck, Germany, Springer, 1998.

[38] MJ. Brusco, D. Steinley *A comparison of heuristic procedures for minimum within-cluster sums of squares partitioning*, Psychometrika, 72, 583-600, 2007.

[39] M.J. Brusco, H.F. Kaohn *Clustering qualitative data based on binary equivalence relations: neighborhood search heuristics for the clique partitioning problem*, Psychometrika, 74, 685-703, 2009.

[40] M. Budinich *Exact bounds on the order of the maximum clique of a graph*, Discrete Applied Mathematics, 127, 535-543, 2003.

[41] M. Budinich, P. Budinich *A Clifford algebra formulation of the maximum clique problem of a graph*, Discrete Applied Mathematics, 145, 52-71, 2004.

[42] M. Budinich, P. Budinich *A Spinorial Formulation of the Maximum Clique Problem of a Graph*, Journal of Mathematical Physics, 47, 430-502, 2006.

[43] T.N. Bui, P.H. Eppley *A hybrid genetic algorithm for the maximum clique problem*, Proc. 6th Int. Conf. Genetic Algorithms, 478-484, 1995.

[44] S. Burer, R.D.C. Monteiro, Y. Zhang *Maximum stable set formulations and heuristics based on continuous optimization*, Mathematical Programming, 137–166, 2002.

[45] S. Busygin *A new trust region technique for the maximum weight clique problem*, Discrete Appl. Math., 154, 2080-2096, 2006.

[46] S. Butenko *Maximum Independent Set and Related Problems with Applications*, A Dissertation presented to the graduate school of the University of Florida in partial fulfillment of the requirements for the degree of the Doctor of Philosophy,University of Florida, 2003.

[47] S. Butenko, P.M. Pardalos *Numerical Methods and Optimization: An Introduction*, Chapman and Hall CRC Press, U.S., 2008.

[48] S. Butenko, O. Yezerska, B. Balasundaram *Variable Objective Search*, Journal of Heuristics, Springer, 1-13, 2011.

[49] SAS Campus *SAS/STAT 9.2 User 's Guide*, SAS Institute Inc., USA, 2008.

[50] Y. Caro, Z. Tuza *Improved lower bounds on k-independence*, Journal of Graph Theory, 15, 99-107, 1991.

[51] R. Carraghan, P.M. Pardalos *An exact algorithm for the maximum clique problem*, Operations Research Letters, 375–382, 1990.

[52] A. Ceselli *Two exact algorithms for the capacitated p-median problem*, 4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Society, 1, 319–340, 2003.

[53] A. Ceselli, A. Righini *A branch-and-price algorithm for the capacitated p-median problem*, Networks, 45, 125–142, 2005.

[54] I. Charon, O. Hudry *Noising methods for a clique partitioning problem*, Discrete Applied Mathematics, 154, 754-769, 2006.

[55] W. Cheswick, H. Burch *Internet Mapping Project*, http://www.cs.belllabs.com/who/ches/map/, Accessed March, 2003.

[56] Y.J. Chiang, M.T. Goodrich, E.F. Grove, R. Tamassia, D.E. Vengroff i J.S. Vitter *External-memory graph algorithms*, In Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 139–149, 1995.

[57] F. Chung, L. Lu *The diameter of random sparse graphs*, Advances in Applied Math., 26, 257–279, 2001.

[58] *http://cucis.ece.northwestern.edu/projects/MAXCLIQUE/download.html*

[59] L. Comtet *Advanced Combinatorics*, Mathematics of Computation, 30(133), 190-192, 1976.

[60] S.A. Cook *The complexity of theorem–proving procedures*, Proc. 3rd Ann. ACM Symp on Theory of Computing, ACM. New York, 151–158, 1971.

[61] C. Cooper, A. Frieze *The size of the largest strongly connected component of a random graph with a given degree sequence*, http://www.math.cmu.edu/~af1p/papers.html, 2002.

[62] P. Crescenzi, C. Fiorini, R. Silvestri *A note on the approximation of the MAX CLIQUE problem*, Inform. Process. Lett., 40, 1-5, 1991.

[63] D. Cvetković, V. Kovačević-Vujčić *Kombinatorna optmizacija*, Društvo operacionih istraživača Jugoslavije, Beograd, 1996.

[64] Dash Optimization Xpress, http://www.dashoptimization.com, Accessed July 2003.

[65] M.S. Daskin *Network and Discrete Location*, J. Wiley and Sons, New York, 1995.

[66] T. Davidović, P. Hansen, N. Mladenović *Permutation based genetic, tabu and variable neighborhood search heuristics for multiprocessor scheduling with communication delays*, Asia-pacific Journal of Operational Research, 22, 297-326, 2005.

[67] T. Davidović *Rasporedjivanje zadataka na višeprocesorske sisteme primenom metaheuristika*, Doktorska disertacija, Univerzitet u Beogradu, Matematički fakultet, Beograd, 2006.

[68] T. Davidović, T.G. Crainic *Benchmark problem instances for static task scheduling of task graphs with communication delays on homogeneous multiprocessor systems*,Comput. OR, 33, 2155-2177, 2006.

[69] T. Davidović, S. Janićijević *Heuristic Approach to Scheduling Independent Tasks on Identical Processors*, XV konferencija YUInfo, Kopaonik, 2008.

[70] T. Davidović, S. Janićijević *VNS for Scheduling Independent Tasks to Identical Processors*, SYMOPIS, 2009.

[71] T.A. Davis, Y. Hu *The university of florida sparse matrix collection*, ACM Transactions on Mathematical Software (TOMS), 38, 1-25, 2011.

[72] S.G. De Amorim, J.P. Barthelemy, C.C.Ribeiro *Clustering and clique partitioning: Simulated annealing and tabu search approaches*, Journal of Classification, 9, 17–41, 1992.

[73] J. Desrosiers, N. Mladenović, D. Villeneuve *Design of balanced MBA student teams*, Journal of the Operational Research Society, 56, 60–66, 2005.

[74] R. Diestel *Graph Theory, Second Edition*, Springer-Verlag, New York, 2000.

[75] C. Ding, C. Wang, Q. Yang, S. Holbrook, *Computing Protein Interaction Modules via Clique Finding based on Generalized Motzkin-Strauss Formalism*, Lawrence Berkeley National Laboratory documentation, Berkeley, 1-4, 2007.

[76] M. Dražić, V. Kovačević-Vujčić, M. Čangalović, N. Mladenović *GLOB — A new VNS-based Software for Global Optimization*, Nonconvex Optimization and Its Applications, 84, 135-154, 2006.

[77] D.Z. Du, B. Gao, W. Wu *A special case for subset interconnection designs*, Discr. Appl. Math., 78, 51-60, 1997.

[78] J. Eisner *State-of-the-Art Algorithms for Minimum Spanning Trees*, Tutorial Discussion, Department of Computer and Information Science, University of Pennsylvania 1997.

[79] P. Erdos, A. Renyi *On random graphs*, Publicationes Mathematicae, 6, 290– 297, 1959.

[80] P. Erdos, A. Renyi *On the evolution of random graphs*, Publ. Math. Inst. Hungar. Acad. Sci., 5, 17–61, 1960.

[81] M. Faloutsos, P. Faloutsos, C. Faloutsos. *On power-law relationships of the Internet topology*, CM-SIGCOMM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication, Cambridge, MA, 29, 251–262, 1999.

[82] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, M. Szegedy *Approximating the maximum clique is almost NP-complete*, Proc. 32nd IEEE Symp. on Foundations of Computer Science, 2-12, 1991.

[83] U. Feige, J. Kilian *Two protocols low error at affordable rates*, Proc. 26. Ann. ACM Symp., Theory of Comput., 172-183, 1994.

[84] T.A. Feo, M.G.C. Resende *A greedy randomized adaptive search procedure for maximum independent set*, Operations Research, 860–878, 1994.

[85] T.A. Feo, M.G.C. Resende *Greedy randomized adaptive search procedures*, Journal of Global Optimization, 109–133, 1995.

[86] A. Frieze *On the independence number of random graphs*, Disctere Mathematics, 81, 171–175, 1990.

[87] K. Fujisawa, M. Kojima, K. Nakata *Exploiting sparsity in primal-dual interior-point methods for semidefinite programming*, Math. Programming, 79, 235-253, 1997.

[88] T. Gallai *Uber extreme Punkt und Kantenmengen*, Ann. Univ. Sci., Eotvos Sect. Math., Budapest, 133–138, 1959.

[89] E. Gardiner, P. Artymiuk, P. Willett *Clique-detection algorithms for matching three-dimensional molecular structures*, Journal of Molecular Graphics and Modelling, 245–253, 1997.

[90] E. Gardiner, P. Willett, P. Artymiuk *Graph-theoretic techniques for macromolecular docking*, J. Chem. Inf. Comput., 40, 273–279, 2000.

[91] M.R. Garey, D.S. Johnson *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman Co. New York, USA, 1979.

[92] L.E. Gibbons, D.W. Hearn, P.M. Pardalos *A continuous based heuristic for the maximum clique problem*, Second DIMACS Implementation Challenge, 103–124, 1996.

[93] L.E. Gibbons, D.W. Hearn, P.M. Pardalos, M.V. Ramana *Continuous characterizations of the maximum clique problem*, Math. Oper. Res., 22, 754–768, 1997.

[94] F. Glover *Tabu search-Part I*, J. Comput 1, 190-260, 1989.

[95] F. Glover, M. Laguna, R. Marti *Fundamentals of scatter search and path relinking*, Control and Cybernetics, 653–684, 2000.

[96] F. Glover, G. Kochenberger, editors *Handbook Of Metaheuristics*, Springer, Boston: Kluwer, 2003.

[97] M.X. Goemans, D.P. Williamson *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of ACM, 1115–1145, 1995.

[98] M.X. Goemans *Semidefinite programming in combinatorial optimization*, Math. Programming, 79, 143-161, 1997.

[99] D.E. Goldberg *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Longman Publishing Co., Boston, USA, 1989.

[100] O. Goldreich *Property testing in massive graphs*, Handbook of Massive Data Sets, Kluwer Academic Publishers, Dordrecht, The Netherlands, 123–147, 2002.

[101] R.E. Gomory *An algorithm for integer solutions to linear programs*, (R. L. Graves and P.Wolfe, editors) Recent Advances in Mathematical Programming, New York, 269-302, 1963.

[102] T. Grossman *Applying the INN model to the max clique problem*, ASIS Symp., 125-146, 1996.

[103] A. Grosso, M. Locatelli, F.D. Croce *Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem*, J. Heuristics, 10, 135-152, 2004.

[104] M. Grotschel, L. Lovasz, A. Schrijver *Polynomial algorithms for perfect graphs*, Ann. Discr. Math., 21, 325-356, 1989.

[105] M. Grotschel, O. Holland *Solution of large-scale travelling salesman problems*, Mathematical Programming, 51, 141-202, 1991.

[106] M. Grotschel, L. Lovasz, A. Schrijver *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 2. edition, 1993.

[107] K. Gursoy *U.T.S. Telekom Istanbul, Turkey: Customer churn analysis in telecommunication sector (based on genetic algorithm for telecommunication customer subdivision*, Istanbul University Journal of the School of Business Administration, 39, 35-49, 2010.

[108] P. Hansen, B. Jaumard *Algorithms for the maximum satisfiability problem*, Computing 44, 279-303, 1990.

[109] P. Hansen, N. Mladenović *Variable neighborhood search*, Computers and Operations Research, 24, 1097-1100, 1997.

[110] P. Hansen, N. Mladenović *Variable neighborhood search: Principles and applications*, European Journal of Operational Research, 130, 449-467, 2001.

[111] P. Hansen, N. Mladenović *J-Means: a new local search heuristic for minimum sum-of-squares clustering*, Pattern Recognit, 34, 405-413, 2001.

[112] P. Hansen, N. Mladenović, D. Urošević *Variable neighborhood search for the Maximum clique*, Discrete Applied Mathematics, 145, 117–125, 2004.

[113] P. Hansen, N. Mladenović, J. A. Moreno Perez *Variable neighbourhood search: algorithms and applications*, Annals of Operations Research, 175, 367-407, 2008.

[114] P. Hansen, N. Mladenović, J. M. Perez *Variable neighbourhood search: methods and applications*, 4OR, 6, 319-360, 2008.

[115] P. Hansen, N. Mladenović, J. Brimberg, J. A. Moreno Perez *Handbook of metaheuristics, chapter Variable Neighbourhood Search*, Kluwer, 2. edition, 146, 61-86, 2010.

[116] J. Harant, A. Pruchnewski, M. Voigt *On dominating sets and independent sets of graphs*, Combinatorics, Probability and Computing, 8, 547–553, 1999.

[117] F. Harary, I.C. Ross *A procedure for clique detection using the group matrix*, Sociometry, 205–215, 1957.

[118] F. Harary *Graph Theory*, Addison-Wesley Publishing Co., 1969.

[119] E. Harley, A. Bonner, N. Goodman *Uniform integration of genome mapping data using intersection graphs*, Bioinformatics, 17, 487–494, 2001.

[120] J. Hastad *Clique is hard to approximate within* $n^{()}$, Acta Mat., 182, 105-142, 1999.

[121] C. Helmberg, F. Rendl, R.J. Vanderbei, H. Wolkowicz *An interior-point method for semidefinite programming*, SIAM J. Optim., 6, 342-361, 1996.

[122] A. Hertz, M. Plumettaz, N. Zufferey *Variable Space Search for Graph Coloring*, Discrete A. Math, 156, 2551-2560, 2007.

[123] S. Hettich, M. Pazzani *Mining for Proposal Reviewers: Lessons Learned at the National Science Foundation*, The Twelfth Annual SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, 862-871, 2006.

[124] M. Hifi *A genetic algorithm - based heuristic for solving the weighted maximum independent set and some equivalent problems*, J. Oper. Res. Soc., 612–622, 1997.

[125] S. Homer, M. Peinado *Experiments with polynomial-time clique approximation algorithms on very large graphs*, DIMACS Series, American Mathematical Society, Providence, RI, 26, 147–167, 1996.

[126] J.J. Hopfield, D.W. Tank *Neural computation of decisions in optimization problems*, Biological Cybernetics, 141–152, 1985.

[127] R.A. Horn, C.R. Johnson *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[128] R. Horst, P. M. Pardalos, N. V. Thoai *Introduction to Global Optimization*, Kluwer Academic Publishers, Springer Science and Business Media, Dordrecht, The Netherlands, 2. edition, 2000.

[129] ILOG CPLEX, http://www.ilog.com/products/cplex/, Accessed July 2003.

[130] J. Jagerskupper *How the (1+1) ES using isotropic mutations minimizes positive definite quadratic forms*, Theoretical Computer Science, 361, 38–56, 2006.

[131] A. Jagota *Approximating maximum clique with a Hopfield network*, IEEE Trans, Neural Networks, 724–735, 1995.

[132] A. Jagota, L. Sanchis, R. Ganesan *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of DIMACS Series, American Mathematical Society, Providence, RI, 169–204, 1996.

[133] S. Janićijević, Z. Lužanin, S. Pereverzyev, I. Stojkovska, A. Tepavčević *Credit score card for corporate clients based on industries*, Working Group ESGI99, Novi Sad, 1-10, 2014.

[134] S. Janićijević, N. Mladenović, D. Urošević *Komparativna analiza klasterovanja na osnovu SAS/STAT procedura, metode promenljivih okolina nad podacima iz baze Telekom Srbija*, SYMOPIS 2014, Divcibare, 2014.

[135] S. Janićijević, N. Mladenović, D. Urošević *Komparativna analiza klasterovanja na osnovu SAS/STAT procedura, metode promenljivih okolina nad podacima iz baze Telekom Srbija*, ITIS 2014, Slovenia, 2014.

[136] S. Janićijević, N. Mladenović, R. Obradović, D. Urošević *VNS for Maximum vertex weighted Clique Problem Approximation*, SYMOPIS 2015, Srebrno jezero, 2015.

[137] S. Janićijević, N. Mladenović, R. Obradović *Fast Exact Higher Order Algorithm for Maximum Clique Problem*, trenutno na recenziji

[138] S. Janićijević, N. Mladenović, R. Obradović *Fast Heuristics Higher Order Algorithm for Maximum Clique Problem*, trenutno na recenziji

[139] S. Janićijević, N. Mladenović, R. Obradović *Formulation Space Search Higher Order Algorithm for Maximum Clique Problem*, trenutno na recenziji

[140] A. Jansen *S.M.H. A Vodafone Case Study*, Vodafone Co., 2007.

[141] S. Janson, T. Luczak, A. Rucinski *Random Graphs*, J. Wiley and Sons, New York, 2000.

[142] M. Jerrum *Large cliques elude the Metropolis Process*, Random Structures and Algorithms 3, 347-359, 1992.

[143] D.S. Johnson, Approximation algorithms for combinatorial problems, Comput. Syst. Sci., 256–278, 1974.

[144] D.S. Johnson, M. Yannakakis, C.H. Papadimitriou *On generating all maximal independent sets*, Inform. Proc. Lett., 27, 119-123, 1988.

[145] D.S. Johnson, M.A. Trick *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, American Mathematical Society, Providence, RI, 26, 619-652, 1996.

[146] R.M. Karp *Reducibility Among Combinatorial Problems*, Complexity of Computer Computations, Springer, New York, 85-103, 1972.

[147] K. Katayama, A. Hamamoto, H. Narihisa*Solving the maximum clique problem by k-opt local search*, Proceedings of the 2004 ACM Symposium on Applied Computing, 1021-1025, 2004.

[148] M.Y. Kiang, M.Y. Hu, D.M. Fisher *American Telephone and Telegraph Company (ATT): An extended self-organizing map (SOM) network for market segmentation*, Decision Support Systems, 42, 36-47, 2006.

[149] V. Klee, D. Larman *Diameters of random graphs*, Canadian Journal of Mathematics, 33, 618–640, 1981.

[150] D.E. Knuth *The sandwich theorem*, Electronic Journal of Combinatorics, Stanford University, 1, 1993.

[151] H. Konno, P. T. Thach, T. Hoang *Optimization on low rank nonconvex structures*, Kluwer Academic Springer US, 15, 95-117, 1997.

[152] Y. Kochetov, P. Kononova, M. Paschenko*Formulation space search approach for the teacher/class timetabling problem*, Yugoslav Journal of Operations Research, 18, 1-11, 2008.

[153] R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins *Trawling the Web for cyber communities*, Computer networks, 31, 1481-1493.

[154] M. Labbe, H. Yaman, E. Gordiny *A Branch and Cut Algorithm for Hub Location Problems with Single Assignment*, Mathematical programming, 102(2), 371-405, 2005.

[155] A.H. Land, A.G. Doig *An automatic method of solving discrete programming problems*, Econometrica, 28, 497-520, 1960.

[156] S. Lawrence, C.L. Giles *Accessibility of information on the Web*, Nature, 400, 107–109, 1999.

[157] C.E. Leiserson, T.B. Schardl *A Work-Efficient Parallel Breadth-First Search Algorithm (or How to Cope with the Nondeterminism of Reducers)*, ACM Symp. on Parallelism in Algorithms and Architectures, 303-314, 2010.

[158] V. Lotfi, R. Cerveny *A final exam scheduling package*, Journal of the Operational Research Society, 42, 205-216, 1991.

[159] L. Lovasz *On the shannon capacity of a graph*, IEEE Trans, Inform. Theory, 1–7, 1979.

[160] T. Luczak *Random trees and random graphs*, Random Structures and Algorithms, 13, 485–500, 1998.

[161] R.N. Mantegna, H.E. Stanley *An Introduction to Econophysics: Correlations and Complexity in Finance*, Cambridge University Press, Cambridge, UK, 1999.

[162] E. Marchiori *A simple heuristic based genetic algorithm for the maximum clique problem*, Proc. ACM Symp. Appl. Comput., 366–373, 1998.

[163] E. Marchiori *Genetic, Iterated and Multistart Local Search for the Maximum Clique Problem*, Proceedings of the Applications of Evolutionary Computing on EvoWorkshops, 112-121, 2002.

[164] P.M. Marcus *Derivation of maximal compatibles using Boolean algebra*, IBM J. Res. Develop., 537–538, 1964.

[165] D. Matula *On the complete subgraph of a random graph*, In R. Bose and T. Dowling, editors, Proceedings of Second Chapel Hill Conference on Combinatorial Mathematics and Its Applications, University of North Carolina, Chapel Hill, 356–369, 1970.

[166] A. Mendelzon, G. Mihaila, T. Milo *Querying theWorld WideWeb*, Journal of Digital Libraries, 1, 68–88, 1997.

[167] J.E. Mitchell *Branch-and-Cut Algorithms for Combinatorial Optimization Problems*, Handbook of Applied Optimization, Oxford University Press, 65-77, 2000.

[168] N. Mladenović *A Variable neighborhood algorithm - a new metaheuristic for combinatorial optimization*, Abstracts of papers presented at Optimization Days, Montreal, 112, 1995.

[169] N. Mladenović, P. Hansen *Variable neighborhood search*, Computers and Operations Research, 24, 1097-1100, 1997.

[170] N. Mladenović, D. Urošević *Variable neighborhood search for the k-cardinality tree*, in Mauricio G. C. Resende and Jorge Pinho de Sousa, editors, Metaheuristics: Computer Decision-Making, Combinatorial Optimization Book Series, Montreal, 481-500, 2003.

[171] N. Mladenović, F. Plastria, D. Urošević *Reformulation descent applied to circle packing problems*, Computers & Operations Research, 32, 2419-2434, 2005.

[172] N. Mladenović *Formulation Space Search - a new approach to optimization (plenary talk)*, SYMOPIS 2005, Vrnjačka Banja, Serbia, 2005.

[173] N. Mladenović, F. Plastria, D. Urošević *Stochastic formulation space search methods*, Proceedings of EURO XXI, Reykjavik, Iceland, 2006.

[174] N. Mladenović, J. Brimberg, J. Hansen, J.M.Perez *The p-median problem: A survey of metaheuristic approaches*, European Journal of Operational Research, 179, 927-939, 2007.

[175] N. Mladenović, M. Dražić, V. Vujčić, M. Čangalović *General variable neighborhood search for the continuous optimization*, European Journal of Operational Research, 191, 753–770, 2008.

[176] M. Molloy, B. Reed *A critical point for random graphs with a given degree sequence*, Random Structures and Algorithms, 6, 161–180, 1995.

[177] J.W. Moon, L. Moser *On cliques in graphs*, Israel Journal of Mathematics, 23–28, 1965.

[178] T. S. Motzkin, E. G. Straus *Maxima for graphs and a new proof of a theorem of Turan*, Canad. J. Math., 17, 533-540, 1965.

[179] M. Negreiros, A. Palhano *The capacitated centred clustering problem*, Comput Oper Res, 33, 1639-1663, 2006.

[180] G.L. Nemhauser, L.E. Trotter *Properties of vertex packings and independence system polyhedra*, Math. Programming, 6, 48-61, 1974.

[181] G.L. Nemhauser, L.E. Trotter *Vertex packings: Structural properties and algorithms*, Math. Programming, 8, 232-248, 1975.

[182] P.R.J. Ostergard *A fast algorithm for the maximum clique problem*, Discrete Applied Mathematics, 197–207, 2002.

[183] M. Padberg, G. Rinaldi *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, DIAM Review, 33, 60-100, 1991.

[184] A. Panconesi, D. Ranjan *Quantifiers and approximation*, Theor. Comput. Sci., 107, 145-163, 1993.

[185] C.H. Papadimitriou, K. Steiglitz *Combinatorial Optimization: Algorithms and Complexity*, Courier Corporation, 1982.

[186] C. Papadimitriou, M. Yannakakis *Optimization, approximation and complexity classes*, Proc. 20. Ann. ACM STOC, 229-234, 1988.

[187] C. Papadimitriou, M. Yannakakis *Optimization, approximation and complexity classes*, J. Comput. Syst. Sci., 43, 425-440, 1991.

[188] P.M. Pardalos, G.P. Rodgers *A Branch and Bound Algorithm For the Maximum Clique Problem*, Computers and Operations Research, 19, 363-375, 1992.

[189] P. Pardalos, J. Xue *The Maximum Clique Problem*, Journal of Global Optimization, 4, 301-328, 1994.

[190] P. Pardalos, D. Du *Handbook of Combinatorial Optimization - The Maximum Clique Problem*, Kluwer Academic Publishers, 1999.

[191] P. Pardalos *Recent Advances in Global Optimization*, Center for Applied Optimization, University of Florida, 1999.

[192] P. M. Pardalos, H. E. Romeijn *Handbook of global optimization*, Kluwer Academic Publishers, University of Florida, Gainesville, Volume 2, 2002.

[193] E. G. Pardo, N. Mladenović, J.J. Pantrigo, A. Duarte *Variable Formulation Search for the Cutwidth Minimization Problem*, Applied Soft Computing, 13, 2242-2252, 2013.

[194] R. Pastor-Satorras, A. Vazquez, A. Vespignani *Dynamical and correlation properties of the Internet*, Physical Review Letters, 87, 658-701, 2001.

[195] B. Pattabiraman, M.M.A. Patwary, A.H. Gebremedhin, W.K. Liao, A. Choudhary*Fast Algorithms for the Maximum Clique Problem on Massive Sparse Graphs*, Optimization Methods and Software, 1–14, 2012.

[196] M.C. Paull, S.H. Unger *Minimizing the number of states in incompletely specified sequential switching functions*, IRE Transactions Electr. Comput., 356–367, 1959.

[197] W. Pullan *Approximating the maximum vertex/edge weighted clique using local search*, J. Heuristics, 14, 117-134, 2008.

[198] W. Pullan, F. Mascia, M. Brunato *Cooperating local search for the maximum clique problem*, J. Heuristics, 17, 181-199, 2011.

[199] S. Regnier *Sur quelques aspects mathématiques des problèmes de classification automatique*, I.C.C. Bulletin, 4, 175–191, 1965.

[200] J.M. Robson *Algorithms for maximum independent sets*, Journal of Algorithms, 425–440, 1986.

[201] R. Samudrala, J. Moult *A graph theoretic algorithm for comparative modeling of protein structure*, J. Mol. Biol., 279, 287–302, 1988.

[202] S. Shakeri, R. Logendran *A mathematical programming based scheduling framework for multitasking environments*, Europ. J. Oper. Res., 209, 176-193, 2007.

[203] A. Schrijver *A course in Combinatorial Optimization*, CWI, Amsterdam, 2012.

[204] N.Y. Shor *Dual quadratic estimates in polynomial and boolean programming*, C.M. in global optimization, 25, 163-168, 1990.

[205] V. Sos, E.G. Straus *Extremal of functions on graphs with applications to graphs and hypergraphs*, J. Combin. Theory, 63, 189–207, 1982.

[206] A. S. Strekalovsky, A. V. Orlov *A new approach to nonconvex optimization*, Numerical Methods and Programming, 8, 11-27, 2001.

[207] A. S. Strekalovsky, A. A. Kuznetsova *On solving the Maximum Clique Problem*, Journal of Global Optimization, 21, 265-288, 2001.

[208] A. S. Strekalovsky, T.V. Gruzdeva *Local Search in Problems with Nonconvex Constraints*, Computational Mathematics and Mathematical Phisics, 47, 381-396, 2001.

[209] A. S. Strekalovsky *Elements of Nonconvex Opti- mization*, Nauka, Novosibirsk, 2003.

[210] A. Struyf, M. Hubert, P.J. Rousseeuw *Clustering in an Object-Oriented Enviroment*, Journal of Statistical Software, 1, 1-30, 2010.

[211] R.E. Tarjan, A.E. Trojanowski *Finding a maximum independent set*, SIAM Journal of Computing, 537–546, 1977.

[212] E. Tomita, A. Tanaka, H. Takahashi *The worst-case time complexity for generating all maximal cliques and computational experiments*, Theoretical Computer Science, 363, 28-42, 2006.

[213] E. Tomita, S. Mitsuma, H. Takahashi *Two algorithms for finding a near-maximum clique*, Tech. Rep. UECTRC1, 1988.

[214] P. Turan *On an extremal problem in graph theory*, Math. Fiz. Lapok, 48, 436-452, 1941.

[215] J.D. Ullman, M. Yannakakis *The input/output complexity of transitive closure*, Annals of Mathematics and Artificial Intelligence, 3, 331–360, 1991.

[216] D. Urošević *Rešavanje problema optimizacije na grafovima Metodom promenljivih okolina*, Doktorska disertacija, Univerzitet u Beogradu, Matematički fakultet, Beograd, oktobar 2004.

[217] O. Verbitsky *On the hardness of approximating some optimization problems that are supposedly easier that MAX CLIQUE*, Comb. Probab. Comput., 4, 167-180, 1995.

[218] J.S. Vitter *External memory algorithms and data structures: dealing with massive data*, ACM Computing Surveys, 33, 209–271, 2001.

[219] S. Vrećica *Konveksna analiza*, BS Procesor, Matematički fakultet, Beograd, 1993.

[220] H. Wang, T. Obremski, B. Alidaee, G. Kochenberger *Clique partitioning for clustering: a comparison with K-means and latent class analysis*, Communications in Statistics Simulation and Computation, 37, 1-13, 2007.

[221] D. Watts *Small Worlds: The Dynamics of Networks Between Order and Randomness*, Princeton University Press, Princeton, NJ, 1999.

[222] H.S. Wilf *Spectral bounds fot he clique and independence numbers of graphs*, J. Combin. Theory B, 40, 113-117, 1986.

[223] Q. Wu, J.K. Hao, F. Glover *Multi-neighborhood tabu search for the maximum weight clique problem*, Ann Oper Res, 1-24, 2012.

[224] M. Yannakakis *On the approximation of maximum satisfiability*, Proc. Annual ACM-SIAM Symp. Disc. Algorithms, 3. edition, 1992.

[225] X.S. Zhang *Neural Networks in Optimization*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

Spisak tabela

Spisak grafika

Spisak algoritama

Biografija

Stefana Janićijević je rođena u Beogradu 10.02.1979. godine. Nakon osnovne škole, upisala je i završila V beogradsku gimnaziju, prirodno-matematičkog smera i srednju Grafičku školu, smer tipografija. Još od tih godina zainteresovala se za istraživanja u nauci i umetnosti, pa je na Beogradskom Univerzitetu 1999. godine upisala Matematički fakultet, smer Profesor matematike i računarstva gde je diplomirala 2006. godine i 2001. godine je upisala Fakultet dramskih umetnosti, smer Pozorišna i radio produkcija gde je diplomirala sa master diplomom 2008. godine sa radom " Analysis of policies, standards and management in digitization of libraries". Program mastera koji je završila se vodi kao UNESCO Chair in Cultural Policy and Management i dvojezičnog je karaktera, sa zvaničnim jezicima francuski i engleski. Superviziju master rada su podržali profesori dr Milena Dragićević Šešić, Gordana Stokić Simončić i dr Zoran Ognjanović. Program doktorskih studija, Matematika u tehnici (Primenjena matematika) Fakulteta tehničkih nauka, je upisala 2008./2009. godine. Poslednju godinu doktorskih studija i istraživačkog rada provela je na Univerzitetu Ljubljana u Sloveniji kao stipendista Erasmus Mundus Basileus programa u trajanju od 10 meseci gde je stekla korisna poznanstva u nauci i novo iskustvo u optimizaciji. Radno iskustvo je stekla kao UNDP konsultant pri Ministarstvu finansija 2006.-2007. godine na projektima institucionalnih i strateških kapaciteta, zatim u Matematičkom institutu na projektu "Informacione tehnologije u digitalizaciji naučne i kulturne baštine" od 2008.-2012. godine gde je radila kao asistent istraživač. Od 2012.- do sada je zaposlena kao stručni saradnik u kompaniji Telekom Srbija u odeljenju za razvoj analitičkih modela za privatne korisnike sektora Poslovne analize za privatne korisnike. Poslovi kojima se aktivno bavi su statistička analiza, razvoj modela, segmentacija, predikciona analiza, vremenske serije, itd.