УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Стеван Љ. Кордић

МЕТОД СЕДИМЕНТАЦИЈЕ И ЊЕГОВЕ ПРИМЈЕНЕ У ПРОБЛЕМИМА ДИСКРЕТНЕ МАТЕМАТИКЕ

докторска дисертација

Београд, 2016

UNIVERSITY OF BELGRADE
FACULTY OF MATHMEMATICS

Stevan Lj. Kordić

METHOD OF SEDIMENTATION AND ITS DISCRETE MATHEMATICS APPLICATIONS

Doctoral dissertation

Belgrade, 2016

О МЕНТОРУ И ЧЛАНОВИМА КОМИСИЈЕ

Ментор:

др Жарко Мијајловић
редовни професор
Математички факултет, Универзитет у Београду

Чланови комисије:

др Татјана Давидовић
виши научни сарадник
Математички институт САНУ

др Предраг Јаничић
редовни професор
Математички факултет, Универзитет у Београду

Датум одбране:

ЗАХВАЛНОСТ

Посебну захвалност дугујем свом ментору *др Жарку Мијајловићу,* без чијег стрпљења и упорност ова теза не би била завршена. Моја захвалност је тим већа, јер сам имао прилике да од њега учим математику, и не само математику, током дугог низа година нашег познанства.

Члану комисије *др Татјани Давидовић,* захваљујем на то што ме увела у свијет проблема оптимизације, као и на неуморној несебичности да своја знања подјели.

Члану комисије, *др Предрагу Јаничићу,* захвалан сам на доброј вољи и њему својственом осјећају за мјеру, који је био више него од помоћи у уобличавању ове тезе.

Желим да се посебно захвалим колегиници *мр Наташи Ковач,* са којом сам заједно упознавао особине проблема додјеле везова бродовима. Њени коментари и расправе, које смо имали у вези програмских рјешења биле су више него драгоцјене за стварање тезе.

Захвалност на разумијевању и спремности да помогну дугујем и колегама *др Ромеу Мештровићу* и *др Браниславу Драговићу*, као и свим колегама са *Поморског факултета у Котору*.

Тезу посвећујем својој фамилији, покојном *оцу Љубу*, *мајци Катици* и *брату Дејану*.

МЕТОД СЕДИМЕНТАЦИЈЕ И ЊЕГОВЕ ПРИМЈЕНЕ У ПРОБЛЕМИМА ДИСКРЕТНЕ МАТЕМАТИКЕ

РЕЗИМЕ

Проблеми задовољења ограничења, којима припадају и *проблеми оптимизације,* спадају у веома значајне проблеме дискретне математике са широким пољем употребе у математици и њеним примјенама.

Дисертација разматра проблем оптимизације и предлаже оригиналан метод за његово тачно рјешавање. Назив алгоритма који се приказује у тези је *метод седимнетације* и приказан је заједно са двије своје хеуристике. Спада у класу алгоритама *„гранања и ограничавања“,* који користе механизме *„враћања по трагу“* и *„провјере унапријед“.* Доказана је његова тотална коректност методе седиментације.

Као илустрација примијењљивости *методе седиментације,* у дисертацији су разматране примјене ове методе на проблеме исказне задовољивости, Вајтхедов проблем минимизације и проблем додјеле везова бродовима у контејнерским лукама. Најбоље резултате метода је показала рјешавајући проблеме додјеле везова, јер је моделовање овог проблема за рјешавање методом седимнетације укључивало све оптимизацијске технике, којима метода располаже. Такође, за проблем додјеле везове, утврђена је прецизна оцјена комплесности методе седиментације. Експериментални резултати потврђују да метода седиментације може да рјешава проблеме додјеле везова на нивоу најбољих приступа овом проблему.

Кључне ријечи:
проблем задовољења ограничења, проблем оптимизације, дискретна математика, комбинаторна оптимизација, тачно рјешење, алгоритам, буловска задовољивост, Вајтхедов проблем минимизације, проблем додјеле везова

Научна област: Математика

Ужа научна област: Дискретна математика

УДК број:

METHOD OF SEDIMENTATION AND ITS DISCRETE MATHEMATICS APPLICATIONS

ABSTRACT

Constrain satisfaction problems including the *optimisation problems* are among the most important problems of discrete mathematics with wide area of application in mathematics itself and in the applied mathematics.

Dissertation study optimisation problem and presents an original method for finding its exact solution. The name of the method is *Sedimentation Algorithm*, which is introduced together with two heuristics. It belongs to the class of *branch-and-bound* algorithms, which uses *backtracking* and *forward checking* techniques. The Sedimentation Algorithm is proven to be totally correct.

Ability of the Sedimentation Algorithm to solve different type of problems is demonstrated in dissertation by its application on the Boolean satisfiability problems, the Whitehead Minimisation Problem and the Berth Allocation Problem in container port. The best results are obtained for Berth Allocation Problem, because its modelling for Sedimentation Algorithm includes all available optimisation techniques of the method. The precise complexity estimation of the Sedimentation Algorithm for the Berth Allocation Problem is established. Experimental results verify that the Sedimentation Algorithm is capable to solve the Berth Allocation Problem on the *state of art* level.

Key words:

constrain satisfaction problem, optimisation problem, discrete mathematics, combinatorial optimisation, exact solution, algorithm, Boolean satisfiability, Whitehead minimisation problem, berth Allocation problem

Scientific field: Mathematics

Scientific subfield: Discrete mathematics

UDK number:

САДРЖАЈ

СПИСАК СКРАЋЕНИЦА

ЋИРИЛИЧНЕ СКРАЋЕНИЦЕ

ПО – Проблем оптимизације

ЛАТИНИЧНЕ СКРАЋЕНИЦЕ

BAP – Berth Allocation Problem, проблем додјеле везова

BB – *Branch-and-Bound*, метод *„гранања и ограничавања“*

BFA – *Brute Force Algorithm*, алгоритам *„грубе силе“* или алгоритам *„исцрпног претраживања“*

BT – *Backtracking*, метод *„враћања по трагу“*

CSP – *Constrain Satisfaction Problem*, проблем задовољења ограничења

D&C – *Divide and Conquer*, стратегија *„подијели, па владај“*

ERH – *Estimate and Rearrange Heuristic*, хеуристика *„процјени и преуреди“*

FC – *Forward checking*, метод *„провјере унапријед“*

GBBA – *General Branch and Bound Algorithm*, општи *„гранај и ограничи“* алгоритам

Max-SAT – *Maximum Satisfiability Problem*, проблем максималне задовољивости формуле

P Max-SAT – *Partial Maximum Satisfiability Problem*, проблем парцијалне задовољивости формуле

PW Max-SAT – *Partial Weighted Maximum Satisfiability Problem*, проблем парцијалне тежинске задовољивости формуле

SAT – *Boolean Satisfiability Problem*, проблем исказне задовољивости

SEDA – *Sedimentation Algorithm*, метод седиментације

W Max-SAT – *Weighted Maximum Satisfiability Problem*, проблем максималне тежинске задовољивости формуле

WMP – *Whitehead Minimization Problem*, Вајтхедов проблем минимизације

1. УВОД

Проблем задовољења ограничења (CSP), састоји се у одређивању вриједности *промјенљивим одлучивања,* тако да оне задовољавају унапријед дефинисани скуп *ограничења*. Уколико се томе дода још и *функција критеријума,* која слика скуп вриједности промјенљивих одлучивања у скуп (или подскуп) реалних бројева, и захтјев да вриједности промјенљивих одлучивања морају да минимизују или максимизују функцију критеријума, добијамо *проблем оптимизације* (ПО). Рјешење ПО зовемо оптимумом.

Проблеми CSP и ПО имају бројне примјене у математици и примјенама. У оквиру тезе разматра се посебно *проблем исказне задовољивости* (SAT), *Вајтхедов проблем минимизације* (WMP) и *проблем додјеле везова у контејнерским лукама* (BAP). ПО, по својој комплексности, могу да буду сасвим једноставни проблеми или веома компликовани, са експоненцијалном или још вишом комплексношћу.

Два су основна принципа рјешавања ПО: *тачни* (егзактни, потпуни) и *приближни* (непотпуни). Тачни алгоритми врше комплетну претрагу допустивих рјешења и долази до оптимума, док приближни претражују само подскуп скупа допустивих рјешења, и на основу те претраге, дају приближну вриједност оптимума. Што је та вриједност ближа оптимуму, то се такав приближни метод сматра бољим. Због комплексности конкретних ПО, приближни методи су учесталији. То не значи да тачни методи губе на значају. Они често немају употребну вриједност, због временски дугог процеса рјешавања. Међутим, важни су како са теоретског становишта, тако и са становишта утврђивања квалитета приближних метода.

Постоје бројни алгоритми, технике, стратегије, хеуристике и методе за рјешавање ПО. За приступ, који се предлаже у тези, веома су важна сљедећа три метода:

1. Метод *„гранања и ограничавања“*, на енглеском *Branch and Bound* (BB).
2. Метод *„враћања по трагу“,* на енглеском језику *Backtracking* (BT).
3. Метод *„подијели, па владај“,* на енглеском језику *Divide & Conquer* (D&C).

Наведене методе се употребљавају, мање или више успјешно, за рјешавање разних проблема у математици и њеним примјенама, укључујући ПО.

Ефикасно рјешавање ПО, обично подразумијева развој комплексних алгоритама, намјенски створених за рјешавање одређеног типа ПО. Опште методе и алгоритми за рјешавање ПО су релативно ријетки. Такође, због комплексности метода, често се у радовима изостављају докази коректност, рачунајући да је псеудокод на апстрактном нивоу довољан доказ исправности. Проблем за оцјену ефикасности појединих алгоритама за рјешавање ПО представља и неусаглашеност тест примјера или непотпун опис како се дошло до примјера који су тестирани.

1.1. ЦИЉЕВИ ТЕЗЕ

Циљеви ове тезе су:

1. Формулација опште методе за рјешавање проблема ПО, која је названа *метод седиментације*.
2. Опис свих услова за примјену *методе седиментације,* потребних да би се она примијенила за рјешавање конкретних ПО.
3. Детаљан опис општег алгоритма рјешавача ПО базираног на *методи седиментације,* као и његових структура, које омогућавају његов ефектан рад.
4. Ригорозна анализа коректности општег рјешавача ПО, базираног на *методи седиментације.*
5. Илустровање примијенљивости *методе седиментације* на SAT и WMP.
6. Илустровање примијенљивости *методе седиментације* на ВАР, конструкцијом рјешавача базираног на *методи седиментације,* који својом брзином рјешавања може да се пореди са *state-of-art* рјешавачима ВАР.
7. Прецизна оцјена комплексности рјешавача ВАР проблема базираног на *методи седиментације.*

1.2. ДОПРИНОСИ ТЕЗЕ

Оригинални доприноси ове тезе прате наведене циљеве. Дио резултата ове тезе је већ објављен, док су неки њени дјелови пријављени за објављивање или су у процесу припреме за објављивање. Наведимо главне доприносе:

1. Формулација и доказ коректности *методе седиментације* и хеуристике *„процјени и преуеди“*, моделовање ВАР-а за рјешавање *Методом седиментације*, оцјена комплексности за тако развијени рјешавач ВАР-а заједно са експерименталним резултатима објављени су у часопису *Applied Mathematical Modelling,* у раду (Kordić, Davidović, Kovač, & Dragović, 2016).
2. Формулација и доказ коректности за стратегију *„подијели, па владај“,* за *метод седиментације*, приказан је на XII Балканској конференцији за операциона истраживања, БАЛКОР 2016, одржаној у Констанци, Румунија. Рад је касније објављен у *proceedings*-у конференције, који је изашао као посебан број часописа *Naval Academy Scientific Bulletin* (Kordić, Kovač, & Davidović, Divide and Conquer Approach to Discrete Berth Allocation Problem, 2015).
3. Формулација и доказ коректности *методе седиментације,* моделовање Max-SAT проблема за рјешавање *методом седиментације* и експериментални резултати за Max-2SAT и Max-3SAT прихваћени су за штампу за часопис *Mathematica Montisnigri*, раду (Kordić, Application of Sedimentation Algorithm for Solving Max-SAT Problem, 2016).
4. Развијен је рјешавач за ВАР, базиран на *методи седиментације,* писан у програмском језику *C*. Развијени су рјешавачи за ВАР, ВАСАР и Max-SAT у програмском језику *Wolfram Language*. На основу наведених рјешавача добијени су експериментални резултати у тези и радовима.

1.3. ОРГАНИЗАЦИЈА ТЕЗЕ

Потребни појмови

У уводном дјелу тезе формулисани су основни појмови везани за CSP, ПО и ВВ, као и два основна алгоритма и њихова својства за рјешавање ПО.

Преглед постојећих метода

У овом дијелу дат је краћа класификација и преглед метода за рјешавање CSP и ПО.

Метод седиментације

У овом дијелу тезе детаљно је формулисан *метод седиментације*, као и двије његове хеуристике *„процјени и преуеди“* и *„подјели, па владај“*. Затим је доказана тотална коректности *методе седиментације* и наведене двије хеуристике.

Примјена методе седиментације
на проблеме у математичкој логици и алгебри

Као илустрација примјењивости *методе седиментације* на проблеме из домена математичке логике и алгебре, приказано је моделовање проблема: SAT, Max-SAT, W Max-SAT, P Max-SAT, PW Max-SAT, као и моделовање WMP. Представљени су и експериментални резултати за Max-2SAT и Max-3SAT проблеме.

Примјена методе седиментације на проблеме у транспорту

ВАР је изабран као трећи примјер илустрације примјењивости *методе седиментације*. За дискретну и хибридну верзију ВАР-а дата је процјена комплексности са рјешаваче базиране на *методи седиментације*, укључујући и двије његове хеуристике *„процјени и преуеди“* и *„подјели, па владај“*. На крају су дати исцрпни експериментални резултати, који садрже међусобно упоређење 3 верзије рјешавача базираних на *методи седиментације,* затим њихово упоређење са CPLEX програмским пакетом за рјешавање проблема линеарног програмирања. Коначно извршена је упоређење са осталим приступима у рјешавању ВАР-а.

Закључна разматрања

У овом поглављу сумирају се резултати тезе и дискутује могући аспекти даљег наставка рада на усавршавању *методе седиментације*.

2. ПОТРЕБНИ ПОЈМОВИ

Два основна појма које уводимо у оквиру овог поглавља су *проблеми задовољења ограничења (Constraint Satisfaction Problem)* и ПО. Класа проблема које називамо CSP су оквир у којем ћемо разматрати ПО. То су централни појмови ове тезе.

2.1. ПРОБЛЕМ ОПТИМИЗАЦИЈЕ

CSP једноставно можемо дефинисати као проблем проналажења валуације за промјенљиве $X_1, X_2, \dots, X_n$, које узимају вриједности из скупа домена $D_1, D_2, \dots, D_n$, а да при томе валуација задовољавају услове $C_1, C_2, \dots, C_m$. Уколико се овоме дода реална *функција критеријума* $f(X_1, X_2, \dots, X_n)$, и задатак се прошири, тако да се још захтјева да валуација минимизује или максимизује функцију критеријума, добијамо *проблем оптимизације*. Управо је то централни проблем којим се бави овај рад.

Надаље се подразумијева да се ради о *дискретној* врсти ПО, тј. оној у којој су скупови $D_1, D_2, \dots, D_n$ коначни.

У литератури постоје бројне формулације CSP-а (Cvetković, и други, 1996), (Russell & Norvig, 2010), (Vujošević, 2012).

2.1.1. Дефиниција проблема оптимизације

У овом подпоглављу слиједи формулација ПО прилагођена теми коју излажемо. Разлике у односу на устаљену дефиницију је само богатија нотација.

Дефиниција 2.1.1 *– Скуп промјенљивих одлучивања и скуп индекса*

Коначан скуп промјенљивих $X = \{x_1, \dots, x_l\}$ назива се *скупом промјенљивих одлучивања*[1]. Још краће, *промјенљиве одлучивања* називају се *промјенљиве*, када такво означавање не доводи у забуну, у односу на остале промјенљиве, које ћемо користити. Очигледно, кардиналност скупа X је l. За скуп промјенљивих одлучивања X, дефинише се и његов *скуп индекса* $L = \{1, \dots, l\}$.

[1] На енглеском језику *скуп промјенљивих одлучивања* се назива *set of decision variables*.

Дефиниција 2.1.2 *– Скуп домена*

Скуп $Dom = \{D_1, \dots, D_l\}$, такав да за његове елементе важи $(\forall i \in L)\ D_i \neq \emptyset$, назива се *скупом домена*. Свака од промјењљивих x_i, $i \in L$, узима вриједност из одговарајућег скупа D_i.

Подразумијева се да су елементи скупова D_i, за $i \in L$ енумерисани. Означаваћемо их са d_{ij}, гдје први индекс i, означава припадност скупу D_i, а други индекс j, његову нумерацију у оквиру елемената скупа D_i. Начин на који се врши нумерација скупова домена је веома важан и биће предметом пажње у наставку тезе. За сада подразумијевамо произвољну нумерацију ових скупова.

Дефиниција 2.1.3 *– Валуација скупа промјењљивих одлучивања*

Пресликавање $e: X \to D_1 \times D_2 \times \cdots \times D_l$, које свакој промјењљивој x_i, $i \in L$, додјељује $e_i \in D_i$ називамо *валуацијом* и означавамо $e(x_i) = e_i$. Умјесто записа $e(x_i)$, за свако $i \in L$, ради једноставности, користи се искључиво e_i.

Дефиниција 2.1.4 *– Скуп услова*

Скуп формула, предикатског рачуна првог реда $\Phi = \{\varphi_1, \dots, \varphi_k\}$, у којима су промјењљиве из скупа X слободне називају се *скупом услова*[2].

У случају када се жели нагласити да су промјењљиве из скупа X слободне пишемо $\varphi_i(X)$ или $\varphi_i(x_1, \dots, x_l)$, за $i \in L$, односно $\Phi(X)$ или $\Phi(x_1, \dots, x_l)$.

Дефиниција 2.1.5 *– Формула услова*

Формулу $\Psi \equiv \varphi_1 \wedge \dots \wedge \varphi_k$ зовемо *формулом услова*.

Из дефиниције 2.1.4, јасно је да формула Ψ такође формула предикатског рачуна првог реда са слободним промјењљивим из скупа X. Када се жели нагласити, да су промјењљиве из скупа X, слободне у формули Ψ, пишемо $\Psi(X)$ или $\Psi(x_1, \dots, x_l)$.

---

[2] На енглеском језику *скуп услова* се назива *the set of constrains*.

***Дефиниција 2.1.6** – Допустиво рјешење*

За валуацију e кажемо да је *допустиво рјешење*[3] ако задовољава све услове из скупа услова Φ, тј. ако важи $\Psi(e)$, односно $\Psi(e_1, \dots, e_l)$.

***Дефиниција 2.1.7** – Простор рјешења*

Скуп $E = \{e \in D_1 \times D_2 \times \cdots \times D_l \mid \Psi(e)\}$, зовемо *простором рјешења*[4].

Из дефиниције *простора рјешења* E, јасно је да за свако допустиво рјешење e важи $e \in E$. Простор рјешења E, може се схватити као скуп свих могућих допустивих рјешења.

***Дефиниција 2.1.8** – Функција критеријума*

Функција $f: D_1 \times D_2 \times \cdots \times D_l \to \mathbb{R}^+ \cup \{0, +\infty\}$, зовемо *функцијом критеријума*[5].

***Дефиниција 2.1.9** – Релација минимизације-максимизације*

Релацију линеарног уређења $\prec$ над скупом $\mathbb{R}^+ \cup \{0, +\infty\}$, зовемо *релацијом минимизације-максимизације*.

Релација минимизације-максимизације је углавном уобичајена релације $<$ и $>$ над реалним бројевима. Разлог увођења појма *релације минимизације-максимизације* је једноставније формулисање еквивалентних проблема минимизације и максимизације. Такође, без издвојене дефиниције користићемо и проширену релације минимизације-максимизације $\preccurlyeq$.

***Дефиниција 2.1.10** – Проблем оптимизације*

Проблем оптимизације, на основу претходних дефиниција дефинише се као проблем проналажења валуације o, тако да важи:

$$(\forall e \in E)\ f(o) \preccurlyeq f(e). \tag{2.1.1}$$

---

[3] На енглеском допустиво рјешење се назива *feasible solution*.

[4] На енглеском језику *простор рјешења* се назива *solution space*.

[5] На енглеском језику *функција критеријума* се назива *objective function*.

Другим ријечима проблем оптимизације је проблем проналажења могућег рјешења са $\prec$ минималном вриједношћу функције критеријума.

Дефиниција 2.1.11 *– Оптимално рјешење и оптимум*

Рјешење ПО, формулисаног у дефиницији 2.1.10, зове се *оптимално рјешење*, а његова вриједност функције критеријума зове се *оптимум*.

Уколико је скуп могућих рјешења E непразан, оптимално рјешење увијек постоји, при томе оно не мора да буде јединствено.

2.1.2. Рјешавање проблема оптимизације

У овом дијелу описује се најједноставнији алгоритам за рјешавање ПО. Тај алгоритам назива се *алгоритмом „грубе силе“* (*Brute Force Algorithm*, BFA). Такође, у овом дијелу описује се и *основни „гранај и ограничи“ алгоритам* (*General Branch and Bound,* GBBA). Прије самог излагања алгоритама уводе се својства *парцијалне коректности* и *тоталне коректности* алгоритама.

Дефиниција 2.1.12 *– Својства парцијалне коректности и тоталне коректности алгоритама*

За алгоритам се каже да је *парцијално коректан* ако никада не даје погрешан резултат. За алгоритам се каже да је *тотално коректан* ако увијек даје исправан резултат.

Разлика између парцијалне и тоталне коректности је у заустављању. Парцијално коректан алгоритам може да не дође до резултата, док тотално коректан алгоритам увијек долази до резултата. Формална и прецизна дефиниција ова два појма налази се у (Kubiak, Rudzinski, & Sokolowski, 1991).

Алгоритам 2.1.1 *– Алгоритам „Грубе силе“*

Улазни подаци BFA-а су скоро сви појмови које су уведени у претходном подпоглављу: X, Dom, Φ, f и $\prec$. Будући, да се промјењиве одлучивања X, подразумјевају, довољно је прослиједити алгоритму као улазни податак њихов број l. Резултат рада алгоритма је уређена двојка $\langle o, f(o)\rangle$, на чијем првом мјесту је валуација o, која представља *оптимално рјешење,* а на другом *оптимум* $f(o)$, уколико они постоје. Алгоритам је формулисан рекурзивно, ради лакшег излагања. Зовемо га алгоритмом *„грубе силе“*, његова формулација је дата је у псеудокоду 2.1.1.

Псеудокод 2.1.1 *– Алгоритам „грубе силе“ за рјешавање ПО*

```
1  BruteForceAlgorith1(l, Dom, Ψ, f, ≺)
2      procedure ReportSolution()
3          if Ψ(x) and f(x) ≺ minimum then
4            minimum = f(x)
5            o = x
6          endif
7      end
8      procedure FindSolution(θ)
9          for i = 1 to |D_θ| do
10             x_θ = d_θi
11             if θ = l then ReportSolution()
12                       else FindSolution(θ + 1)
13             endif
14         endfor
15     end
16     o = ⟨ ∅ | i = 1, … , l ⟩
17     x = ⟨ ∅ | i = 1, … , l ⟩
18     minimum = +∞
19     FindSolution(1)
20     return ⟨o, minimum⟩
21 end
```

BFA испитује редом све валуације x. Уколико валуација x, јесте допустиво рјешење, упоређује се вриједност његове функције критеријума $f(x)$ са до сада најмањом забиљеженом вриједношћу, која се чува у помоћној промјенљивој $minimum$. Уколико је и овај услов испуњен, x постаје нова вриједност за o, тј. $o = x$ и $minimum = f(x)$. Када се, на овај начин, испитају све могуће валуације, промјенљива o ће садржати *оптимално рјешење* ПО, за
дате улазне параметре, а промјенљива $minimum$ оптимум.

Испитајмо ближе псеудокод BFA, јер је он основа за све остале алгоритме који слиједе у наставку тезе.

Алгоритам има двије процедуре ReportSolution() и FindSolution(θ). Прва провјерава, у линији **(3)**, да ли је валуација x, допустиво рјешење тј. да ли важи $\Psi(x)$. Потом се провјерава да ли је $f(x) \prec minimum$, тј. да ли је за допустиво рјешење x функција критеријума $f(x)$ мања од до сада забиљежене у промјенљивој $minimum$? Уколико јесте, допустиво рјешење x је оно са до

сата најмањом вриједности функције критеријума $f(x)$, и они у линијама (**4**) и (**5**) постају нове вриједности за $minimum$ и o.

Друга, рекурзивна процедура FindSolution(θ), за $\theta \in \{1, \dots, l\}$, испитује једну по једну вриједност промјењиве x_θ, тако што конструише валуацију x, са свим могућим вриједностима скупа домена D_θ, у **for** петљи (**9**)—(**14**). Конструкција се врши, тако што се изврши додјела $x_\theta = d_{\theta 1}$, па пређе на сљедећу промјењиву рекурзивним позивом FindSolution$(\theta + 1)$, линије кода (**10**)—(**12**). Након повратка наставља се са $x_\theta = d_{\theta 2}$, па пређе на сљедећу промјењиву рекурзивним позивом FindSolution$(\theta + 1)$ и све тако, док се не испитају сви елементи скупа домена D_θ. Изузетак од рекурзије настаје када је $\theta = l$, у том случају валуацију x је комплетна, па се онда провјерава да ли је она допустиво рјешење. Ако валуација x јесте допустиво рјешење, провјерава се још да ли је x ново оптимално рјешење процедури ReportSolution(), линија (**11**).

Теорема 2.1.1 *– Теорема о тоталној коректности BFA-а*
BFA је тотално коректан алгоритам, тј. увијек проналази оптимално рјешење ПО.

Доказ.
Рекурзивна процедура FindSolution(θ), у слиједу својих рекурзивних позива: FindSolution(1), FindSolution(2), до FindSolution(l), разматра све могуће валуације $x: X \to D_1 \times D_2 \times \cdots \times D_l$ и прослеђује их процедури ReportSolution(). Докажимо претходно тврђење индукцијом по $\theta \in \{1, \dots, l\}$.

За $\theta = 1$, први елемент валуације x узима вриједности $x_1 = d_{1i}$, за $i \in \{1, \dots, |D_1|\}$, односно све могуће вриједности домена D_1. Дакле све могуће валуације за промјењиву x_1.

Претпоставимо даље да процедура FindSolution(j) разматра све валуације за $j < \theta \leq l$ и докажимо да ће она то исто да уради и за θ. То значи да наведене процедуре стварају све елементе скупа $D_1 \times D_2 \times \cdots \times D_{\theta-1}$. Означимо са $x \in D_1 \times D_2 \times \cdots \times D_{\theta-1}$, било коју валуацију у позиву процедуре FindSolution(θ). Њој додајемо елемент x_θ тако што испробамо све могуће вриједности скупа домена D_θ, тј. $x_\theta = d_{\theta i}$, за $i \in \{1, \dots, |D_\theta|\}$ и настављамо са

рекурзивним позивима процедуре $\text{FindSolution}(\theta)$. На описани начин проширена валуација x је проширена са свим могућим вриједностима x_θ из скупа D_θ. Тако се стварју све (проширене) валуације $x \in D_1 \times D_2 \times \cdots \times D_\theta$, чиме се закључује индуктивни доказ.

Процедура $\text{ReportSolution}()$ прво провјерава да ли је прослијеђена валуација x допустиво рјешење, тј. да ли важи: $\Psi(x)$. Уколико јесте, провјерава да ли је вриједност њене функције критеријума $f(x)$, мања у релацији $\prec$, у односу на најмању до сада испитану, за допустиво рјешење o. тј. да ли важи: $f(x) \prec f(o)$. Ако је и овај услов испуњен, валуација x, постаје ново допустиво рјешење o, са најмањом до сада испитаном вриједности функције критеријума f. **QED**.

Наведену аргументацију можемо сумирати сљедећом формулом:

$$\underbrace{\min_{f,\prec}\left\{o \mid o \in \underbrace{\{\, x \mid x \in D_1 \times D_2 \times \cdots \times D_l \,\}}_{\text{FindSolution}(\theta)} \wedge \Psi(o)\right\}}_{\text{ReportSolution}()} \tag{2.1.2}$$

Теорема 2.1.2 *– Теорема о комплексности BFA-а*

Комплексност BFA-а је:

$$O\left(\prod_{\theta=1}^{l}|D_i|\right) \tag{2.1.3}$$

Доказ.

Доказ непосредно слиједи из чињенице да BFA испитује све могуће валуације $x: X \to D_1 \times D_2 \times \cdots \times D_l$. Њих има онолико, колико има елемената скупа $D_1 \times D_2 \times \cdots \times D_l$, тј. $\prod_{\theta=1}^{l}|D_i|$. Дакле, комплексност BFA-а је $O\left(\prod_{\theta=1}^{l}|D_i|\right)$. **QED**.

Дефиниција 2.1.13 *– Доња апроксимација вриједносне функција*

За дату функцију критеријума $f: D_1 \times D_2 \times \cdots \times D_l \to \mathbb{R}^+ \cup \{0, +\infty\}$, фамилију функција $f_\theta^*: D_1 \times D_2 \times \cdots \times D_\theta \to \mathbb{R}^+ \cup \{0, +\infty\}$, за $1 \leq \theta \leq l$, назива се *доњом апроксимацијом функције критеријума* f, уколико важе услови:

$$(\forall \theta \in \{1, \dots, l\})(\forall x \in D_1 \times D_2 \times \cdots \times D_l)\ f_\theta^*(x) \preccurlyeq f(x). \tag{2.1.4}$$

$$(\forall \theta \in \{1, \dots, l-1\})(\forall x \in D_1 \times D_2 \times \cdots \times D_l)\ \ f_\theta^*(x) \preccurlyeq f_{\theta+1}^*(x). \qquad (2.1.5)$$

Доња апроксимација функције критеријума f_θ^* користи се за процјену минималне могуће вриједности функције критеријума f, ако су одређене вриједности првих $x_1, \dots, x_\theta$ промјенљивих.

Псеудокод 2.1.2 *– Општи алгоритам „гранај и ограничи“ за рјешавање ПО*

```
1  GeneralBranchAndBoundAlgorithm(l, Dom, f, ≺)
2      procedure ReportSolution()
3          if Ψ(x) and f(x) ≺ minimum then
4            minimum = f(x)
5            o = x
6          endif
7      end
8      procedure FindSolution(θ)
9          for i = 1 to |D_θ| do
10             x_θ = d_θi
11             if f*_θ(x_1, ..., x_θ) ≺ minimum then
12                 if θ = l then ReportSolution()
13                         else FindSolution(θ + 1)
14                 endif
15             endif
16         endfor
17     end
18     o = ⟨ ∅ | i = 1, ..., l ⟩
19     x = ⟨ ∅ | i = 1, ..., l ⟩
20     minimum = +∞
21     FindSolution(1)
22     return ⟨o, minimum⟩
23 end
```

Алгоритам 2.1.2 *– Општи алгоритам „гранај и ограничи“*

Уколико је могуће конструисати (нетривијалну) доњу апроксимацију функције критеријума f_θ^*, BFA је могуће убрзати провјеравањем да ли је додјелом вриједности за промјенљиву x_θ вриједност функције $f_\theta^*(x_1, \dots, x_\theta)$ већа од вриједности $minimum$, тј. минимума за тренутно најбоље рјешење o. Овако модификовани алгоритам за рјешавање ПО, зове се *општим „гранај и ограничи“ алгоритмом*, краће се означава са GBBA, дат је у псеудокоду 2.1.2.

Теорема 2.1.3 *– Теорема о коректности GBBA-а*

GBBA је тотално коректан, тј. проналази оптимално рјешење.

Доказ.

Формулација GBBA је веома слична BFA. Псеудокодови ова два алгоритма разликују се само у једној **if** наредби, која се налази у (**11**) линији псеудокода за GBBA. Како је већ показано у теореми 2.1.1, алгоритам BFA је коректан, па је довољно доказати да се коректност не нарушава у наведеној **if** наредби.

У линији (**11**) псеудокода за GBBA испитује се услов:

$$f_\theta^*(x_1, \dots, x_\theta) \prec minimum. \tag{2.1.6}$$

Уколико је услов испуњен, GBBA се понаша једнако као и BFA. Ако услов није испуњен, онда важи:

$$minimum \preccurlyeq f_\theta^*(x_1, \dots, x_\theta). \tag{2.1.7}$$

То значи да је додјелом $x_\theta = d_{\theta i}$, вриједност доње апроксимације функције критеријума $f_\theta^*(x_1, \dots, x_\theta)$, постала већа или једнака од вриједности тренутног могућег рјешења $minimum = f(o)$. Из својстава функције апроксимације, тј. формула 2.1.3 и 2.1.4, лако се закључује да:

$$\begin{aligned}&(\forall x \in \{x_1\} \times \cdots \times \{x_\theta\} \times D_{\theta+1} \times \cdots \times D_l)\\ &\quad minimum = f(o) \preccurlyeq f_\theta^*(x) \preccurlyeq f_{\theta+1}^*(x) \preccurlyeq \cdots \preccurlyeq f_l^*(x) = f(x).\end{aligned} \tag{2.1.8}$$

Из горње формуле (2.1.7) се закључује да додјелом $x_\theta = d_{\theta i}$ није могуће доћи до оптималнијег рјешења од o, па ову додјелу даље не разматрамо, јер не може довести до побољшања тренутно најбољег рјешења o. На основу овога непосредно закључујемо коректност GBBA-а. **QED**.

Квалитет доње апроксимације функције критеријума f_θ^* веома утиче на ефикасност GBBA-а. Уколико су одређене вриједности за промјенљиве $x_1, \dots, x_\theta$, за $\theta \in \{1, \dots, l\}$ и уколико постоји рјешење ПО o', са таквим почетним вриједностима, онда ће квалитет доње апроксимације функције критеријума f_θ^* бити бољи уколико је разлика $f(x) - f_\theta^*(x)$, за било које $x \in \{x_1\} \times \cdots \times \{x_\theta\} \times D_{\theta+1} \times \cdots \times D_l$, мања.

3. ПРЕГЛЕД ПОСТОЈЕЋИХ МЕТОДА

Као што је већ наведено *метод седиментације* рјешава ПО, у оквиру CSP формулације. Три механизма на које се ослања су BT, FC и BB. Имајући у виду општост ових метода, дајемо краћи њихов преглед. Детаље о CSP-у могуће је наћи у (Dechter, 2003), (Freuder & Mackworth, 2006), (Tsang, 1993) и (Vujošević, 2012).

3.1. ПРОБЛЕМИ ЗАДОВОЉЕЊА ОГРАНИЧЕЊА

Проблеми задовољења ограничења предметом су пажње истраживача већ дуги низ година (Tsang, 1993). Поред тога што се сматрају једном од главних техника вјештачке интелигенције, своју практичну примјену налазне у разним областима, као што су: операциона истраживања, биоинформатика, телекомуникације, електротехника и електроника, итд.

Формална дефиниција CSP-а дата је у поглављу 2.1. Рјешење CSP-а не мора да буде јединствено. Стога, приступе рјешавању CSP-а разликујемо на сљедећи начин:

1. CSP рјешавамо тако да тражимо само једно рјешење (од много њих).
2. CSP рјешавамо тако да тражимо сва могућа рјешења.
3. CSP рјешавамо минимизирајући или максимизирајући функцију критеријума. Овај приступ не искључује постојање више рјешења.

CSP проблеме разликујемо и по типу скупа домена за промјенљиве које треба одредити. Углавном разликујемо два типа домена: цјелобројне и реалне. При томе, најчешће су цјелобројни домени коначни.

Ограничења која се јављају у CSP-у разликујемо по броју промјенљивих, који се јављају у њему. Уколико се ради само о једној промјенљивој, говоримо о *унарном* типу ограничења, ако се у ограничењу јављају двије промјенљиве, тада имамо *бинарни* тип ограничење, а ограничења са више од двије промјенљиве зовемо *општим* ограничењима. Формулација ограничења је слична оној у цјелобројном програмирању, са којим CSP показује извјесне сличности.

Приликом рјешавања CSP-а, питање редослиједа испитивања промјењљивих је од кључног значаја. Погрешан избор, може да успори поступак рјешавања, а добар избор може значајно да га убрза. Постоје два принципа избора промјењљивих одлучивања:

1. *статички*, у којем је поредак унапријед одређен и
2. *динамички*, у којем се поредак мијења током процеса рјешавања CSP-а.

За избору промјењљивих у динамичком приступу често се примјењује хеуристика *„мање кардиналности домена“* (*Minimum Remaining Values*, MRS), која бира промјењљиве одлучивања сходно кардиналности њихових домена. Оне промјењљиве, чија је кардиналност домена мања иду испред оних чија је кардиналност домена већа. Ова хеуристика нарочито долази до изражаја код проблема у којима постоји велика разлика у кардиналности домена. Друга хеуристика, која се често примјењује је испитати прво вриједности промјењљивих које се јављају у већем броју услова.

Други поредак, који је важан, је поредак вриједности у скупу домена. Тежи се да се прво отворе оне гране које воде ка рјешењу. Хеуристика *„најмање ограничавајуће вриједности“* (*Least Constraining Value*, LCV) примјењује се да би промјењљивој прво додијелили вриједност, која ће елиминисати најмање вриједности за друге промјењљиве одлучивања, како би усмјерили претраживање ка бржем проналажењу рјешења. Ова хеуристика не мора нужно да води ка бржем рјешавању, нарочито ако рјешавамо ПО.

3.2. ПРЕГЛЕД ПОСТОЈЕЋИХ МЕТОДА ЗА РЈЕШАВАЊЕ ПРОБЛЕМА ОГРАНИЧЕЊА

Основни метод рјешавања CSP-а је претраживање простора допустивих рјешења. Већ смо приказали два основна алгоритма BFA и GBBA у подпоглављу 2.1.2. У наставку слиједи кратак приказ техника и метода, које су релевантне за остатак тезе.

3.2.1. Метод „враћања по трагу“

Метод који најчешће користимо за претраживање простора допустивих рјешења је метод *„враћања по трагу“*, или на енглеском *„Backtracking”* (BT). Ради се о општем алгоритму за рјешавање широке класе проблема, укључујући и CSP. Може се примијенити на проблеме у којима постоји концепт

„парцијалног кандидата за рјешење“, који омогућава да на основу неких одређе-них вриједности за промјењиве одлучивања, дођемо до закључка да ли за те вриједности може да постоји допустиво рјешење (или оптимално рјешење). Према (Freuder & Mackworth, 2006) термин *„backtacking“* први је употребио амерички математича Д. Х. Лехмер 1950. године. Поред CSP-а, употребљава се за рјешавање задатака комбинаторне оптимизације, синтаксну анализу природних и програмских језика, основни је механизам програмских језика *Prolog* и *Planer*. Формална дефиниција и детаљи о методи *„враћања по трагу“* налазе се у (Knuth, 2016).

3.2.2. Метод „провјере унапријед“

Метод *„провјере унапријед“*, на енглеском језику *„Forward checking”* (FC), је такође једна од техника која се користи при рјешавању CSP-а. Састоји се у томе, да сваки пут, када додијелимо вриједност некој промјењивој одлучивања, привремено избришемо све вриједности из домена осталих промјењивих одлучивања, чија вриједност још није испитивана, а која нарушавају неки од услова са посљедњом додјелом. У случају да, због брисања, неки од наведених домена постане празан скуп, тада у тој грани не постоји допустиво рјешење. Тада се методом *„враћања по трагу“* врши повратак уназад. Метод *„провјере унапријед“* омогућава предвиђање додјеле, која не води ка допустивом рјешењу. По свом типу метод *„провјере унапријед“* спада у технике *„гледај унапријед“*, на енглеском језику *„Look ahead”*. Више о овом типу техника налази се у (Dechter, 2003).

3.2.3. Метод „гранај и ограничи“

Метод *„гранај и ограничи“* користи се за рјешавање многих проблема комбинаторне оптимизације. Аутори метода су А. Х. Ланг и А. Г. Доиг у (Land & Doig, 1960), који су га први примијенили у контексту линеарног програмирања 1960. године. Од тада, овај метод један је од оних који се најчешће користи, у комбинацији са другим техникама, за рјешавање тешких ПО. Термин на енглеском *„Branch and Bound”* за овај метод јавља се у раду (Little, Murty, Sweeney, & Karel, 1963), у којем се приказује алгоритам за рјешавање *проблема путујућег трговца*, базираног на ВВ. Најчешће се примјењује у областима: цјелобројног програмирања, нелинеарног програмирања, транспортним проблемима, проблем максималне задовољивости формуле (Max-SAT), проблемима претраживања блиских околина, машинског учења

и многим другим. Такође, ВВ је основа и за многе хеуристике за приближно рјешавање ПО.

4. МЕТОД СЕДИМЕНТАЦИЈЕ

Метод седиментације је општи метод за тачно рјешавање CSP-а. Припада класи метода *„гранај и ограничи“*, на енглеском језику *„Branch and Bound“* (BB) алгоритама. Користит механизам *„враћања по трагу“*, пропагацију, односно механизам *„провјере унапријед“*, као и низ хеуристика, које му омогућавају да тачно ријеши неки од ПО, вршећи комплетну претрагу простора рјешења. CSP може се ријешити *методом седиментације,* уколико га је могуће моделовати за њега. Начин моделовања проблема умногоме одређује ефикасност рјешавања, па је томе потребно посветити посебну пажњу. Такође, *метод седиментације* омогућује, поред употребе хеуристика самог метода, употребу и специфичних хеуристика везаних за конкретан проблем ограничења, којим се додатно смањује претраживање простора допустивих рјешења.

У овом поглављу биће изложен основни *метод седиментације (Sedimentation Algorithm)* заједно са двије хеуристике. У наставку текста *метод седиментације* краће обиљежавмо са четири латинична слова: *SEDA*. Прва његова хеуристика се назива *Процијени и преуреди (Estimate & Rearrange Heuristic,* скраћено ERH*)*. Она прије рада SEDA покушава да процјени, што прецизније, колика би могла да буде вриједност оптимума. Сходно најмањој (или највећој), вриједности преуреди редослијед додјељивања вриједности промјенљивима и потом поново започне систематично претраживање простора рјешења позивајући SEDA. Друга хеуристика примјењује стратегију *„подјели, па владај“ (Divide & Conquer,* скраћено D&C*)*, која омогућава да се проблем разбије на подпроблеме, и када се они ријеше, дође до рјешења полазног проблема. Ова стратегија може се примјењивати и на друге приступе рјешавања *проблема оптимизације*, ако су испуњени услови за њену примјену.

4.1. АЛГОРИТАМ СЕДИМЕНТАЦИЈЕ

У сљедећим подпоглављима формулисан је *метод седиментације*, навођењем његових улазних података и псеудокода. Затим се доказује потпуна коректност алгоритма и аје коментар о његовој општој комплексности.

4.1.1. Улазни подаци за SEDA-а

Улазни подаци за SEDA-а су у складу са дефиниција уведеним у потпоглављу 2.1.1. Ради лакшег праћења, наводе се у краћој форми:

1. $X = \{x_1, \dots, x_l\}$ – коначан *скуп промјењивих одлучивања*.
2. $Dom = \{D_1, \dots, D_l\}$ – *скуп домена* (могућих вриједности) за промјењиве одлучивања, $x_i \in D_i$, за $i \in L$. Подразумијева се да су скупови D_i коначни и непразни за свако $i \in L$.
3. $\Phi = \{\varphi_1, \dots, \varphi_k\}$ – *скуп услова,* које требају да задовољавају вриједности промјењивих одлучивања.
4. f – *функција критеријума,* која зависи од вриједности промјењивих одлучивања, тј. $f(x_1, \dots, x_l)$. Подразумијева се функција критеријума има следећи облик:

$$f(x_1, \dots, x_l) = \sum_{k=1}^{l} f_k(x_k). \tag{4.1.1}$$

   При томе, функције $f_k(x_k)$ су ненегативне за свако $k \in L$.
5. Ξ – *скуп хеуристичких релација* $\Xi = \{\prec_1, \dots, \prec_l\}$ садржи хеуристике релације тоталног уређења $\prec_i \subseteq D_i^2$, за сваки скуп домена D_i. Хеуристичка релација $\prec_i$, за свака два члана $a, b \in D_i$, *„одлучује“* која је вриједност за промјењиву одлучивања x_i боља: a или b. Вриједност a сматрамо *„бољом“* од вриједности b ако $a \prec_i b$. Хеуристичке релације морају да задовољавају услов:

$$(\forall i \in \{1, \dots l\})(\forall a, b \in D_i)\ \ a \prec_i b \Rightarrow\ f_i(a) \leq f_i(b). \tag{4.1.2}$$

6. $F(\theta)$ – *функција завршетка конструкције рјешења* враћа вриједност *тачно* уколико се може доћи до закључка да је оптимално рјешење конструисано на основу θ одређених промјењивих одлучивања. Преосталим $l - \theta$ промјењивим одлучивања се одређује вриједност, или на основу оних већ одређених, или њихова вриједност нема значаја за вриједност функције критеријума, па можемо узети било коју. У супротном $F(\theta)$ враћа вриједност *нетачно*.

4.1.2. Структуре података које користи SEDA

SEDA користи сљедеће структуре података:

1. o – низ података у којем су смјештене вриједности промјењивих одлучивања, које одговарају тренутно најбољем рјешењу проблема оптимизације, који се рјешава.
2. $minimum$ – промјењива у којој се чува вриједност функције критеријума за тренутно најбоље рјешење. Промјењива $minimum$ је у потпуности одређена вриједностима тренутно најбољег рјешења i.e., $minimum = f(o_1, \dots, o_l)$, уколико је o одређено, иначе подразумијева се да је $minimum = +\infty$.
3. θ – бројач промјењивих одлучивања. Бројач θ истовремено означава и број корака у конструкцији допустивих рјешења.

4.1.3. Функције и процедуре које користи SEDA

Функције и процедуре које користи SEDA наводимо у низу подпоглавља, које ће пратити њихов детаљан опис.

***Процедура* 4.1.1** – FindSolution(θ, Dom)

Процедура FindSolution(θ, Dom) је рекурзивна процедура која одређује вриједност за промјењиву одлучивања x_θ, бирајући вриједност из скупа домена $D_\theta \in Dom$. Рекурзија започиње за вриједношћу $\theta = 1$, и наставља се док се не одреди комплетно допустиво рјешење, за $\theta = l,$ или када је испуњен услов завршетка конструкције $F(\theta)$. Ова процедура ће бити детаљније описана у коментарима псеудокода SEDA-а.

***Функција* 4.1.2** – Estimation(θ, Dom)

Функција која процјењује могућу вриједност функције критеријума f. Уколико су вриједност првих $\theta - 1$ промјењивих одлучивања $\{x_1, \dots, x_{\theta-1}\}$ одређене, тада функцију можемо представити као:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \text{NonDetVarEstimation}(\theta). \tag{4.1.3}$$

Горња сума се рачуна као функција критеријума за вриједности промјењивих одлучивања за које је одређена вриједност: $\{x_1, \dots, x_{\theta-1}\}$. За промјењиве одлучивања $\{x_\theta, \dots, x_l\}$, чија вриједност није одређена користи се

функција $\text{NonDetVarEstimation}(\theta)$ за процјену минимална вриједност функције
критеријума f. Квалитет оцјене веома утиче на брзину рада SEDA-а.

Претпоставка коју даље подразумијевати о функцији $\text{Estimation}(\theta, Dom)$ је да је њена вриједност увијек мања или једнака функцији критеријума f за оптимално рјешење, чије су вриједности промјенљивих одлучивања $\{x_1, \dots, x_{\theta-1}\}$ одређене, и да при томе важи:

$$(\forall k \in \{\theta, \dots, l\})\ D'_k \neq \emptyset. \tag{4.1.4}$$

Скупови $D'_k \subseteq D_k$, за $k \in \{\theta, \dots, l\}$, представљају подскупове домена за промјенљиве $\{x_\theta, \dots, x_l\}$ из којих су искључени они елементи који не задовољавају услове Φ, са вриједностима већ одређених вриједности промјенљивих одлучивања $\{x_1, \dots, x_{\theta-1}\}$. Уколико је $D'_k = \emptyset$, за неко $k \in \{\theta, \dots, l\}$, тада не постоји допустиво рјешење са почетним одређеним вриједностима $\{x_1, \dots, x_{\theta-1}\}$. У том случају узимамо да је $\text{Estimation}(\theta, Dom) = +\infty$.

Функција $\text{NonDetVarEstimation}(\theta)$, рачуна се формулом:

$$\text{NonDetVarEstimation}(\theta) = \sum_{k=\theta}^{l} \begin{cases} \min\limits_{y \in D_k} f_k(y) & : \quad D_k \neq \emptyset, \\ +\infty & : \quad D_k = \emptyset. \end{cases} \tag{4.1.5}$$

На основу формуле (4.1.5), функција $\text{Estimation}(\theta, Dom)$ може се представити као:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \sum_{k=\theta}^{l} \begin{cases} \min\limits_{y \in D_k} f_k(y) & : \quad D_k \neq \emptyset, \\ +\infty & : \quad D_k = \emptyset. \end{cases} \tag{4.1.6}$$

За овако формулисану функцију $\text{Estimation}(\theta, Dom)$, све наведене претпоставке су испуњене.

***Функција* 4.1.3** – $\text{Sediment}(\theta, x)$

Функција $\text{Sediment}(\theta, x)$ – *функција таложења* – врши пропагацију додјеле вриједности x промјенљивој одлучивања x_θ у свим доменима $\{D_{\theta+1}, \dots, D_l\}$. Такође, у њој се примјењују разне опште и специфичне *„гледај унапријед“*, методе укључујући и метод *„провјере унапријед“*, у циљу даљег редуковања домена $\{D_{\theta+1}, \dots, D_l\}$, при томе водећи рачуна да се у тим редукцијама не елиминишу вриједности оптималног рјешења проблема. Након *„таложења“*

функција враћа нове вриједности за скуп домена. Псеудокод ове функције је дат у псеудокоду 4.1.1.

Псеудокод 4.1.1 – *Sediment функција за пропагацију и редуковање скупа домена*

```
1  Sediment(θ, x)
2      for i = θ + 1 to l do
3          "Уклонити елементе из D_i који не задовољавају
               услове проблема, ако x_θ = x"
           "Уклонити елементе из D_i који не задовољавају
               специфичне технике 'гледања унапријед', ако x_θ = x"
4      endfor
5      return D
6  end
```

Функција Sediment(θ, x) је инспирисала назив метода, јер подсјећа на појаву таложења честица у течностима.

Процедура 4.1.4 – ReportSolution()

Процедура ReportSolution(), провјерава да ли је ново допустиву рјешење има мању вриједност функције критеријума f од тренутно најбоље забиљежене у o. Уколико је то тачно, онда ново допустиво рјешење постаје најбоље забиљежено и чува се у промјенљивој o, као и вриједност функције критеријума $f(o)$, за нову вриједност o, коју чувамо у промјенљивој $minimum$.

Функција 4.1.5 – CutOff(θ, ε)

Функција CutOff(θ, ε), искључује елементе из скупа домена $\{D_{\theta+1}, \dots, D_l\}$, за промјенљиве $\{x_{\theta+1}, \dots, x_l\}$, којима још није одређена вриједност у току рада SEDA, а имају велику вриједност функција f_k, за $k \in \{\theta + 1, \dots, l\}$.

Објаснимо детаљније који се елементи искључују. Означимо са M_k збир мимималних вриједности функција f_k на домену D_k, за $k \in \{\theta + 1, \dots, l\}$ и позитивног броја ε, тј.,

$$M_k = \min_{y \in D_k} f_k(y) + \varepsilon, \quad k \in \{\theta + 1, \dots, l\}. \tag{4.1.7}$$

Функција CutOff(θ, ε), израчунава нови скуп вриједности скупа домена: $\{D_1, \dots, D_\theta, D'_{\theta+1}, \dots, D'_l\}$. Скуп домена, за промјенљиву чија је вриједност одређена, је непромијењен, док се нове вриједност скупа домена за промјенљиве, чија је вриједност неодређена одређује на сљедећи начин:

$$D'_k = \{\, y \in D_k \mid f_k(y) < M_k \,\}, \quad k \in \{\theta + 1, \ldots, l\}. \tag{4.1.8}$$

***Функција* 4.1.6** – $\mathrm{Minx}(D, \prec)$

Функција $\mathrm{Minx}(D, \prec)$ одређује $\prec$ минимални елемент скупа D, при чему се претпоставља да је $\prec \subseteq D^2$, релација тоталног уређења, односно она одређује $a \in D$, тако да важи:

$$(\forall y \in D)\ y \neq a \ \Rightarrow\ a \prec y. \tag{4.1.9}$$

***Функција* 4.1.7** – $F(\theta)$

Функција $F(\theta)$ враћа вриједност *тачно* уколико можемо доћи до закључка да је оптимално рјешење конструисано на основу θ одређених промјенљивих одлучивања. Преосталим $l - \theta$ промјенљивим одлучивања, одређујемо вриједност на основу оних већ одређених, или њихова вриједност нема значаја за вриједност функције критеријума, па можемо узети било коју. У супротном $F(\theta)$ враћа вриједност *нетачно*.

4.1.4. Имплементација функција и процедура које користи SEDA

Један од главни разлог ефикасности SEDA-а лежи у начину на који су функције и процедуре описане у претходном поглављу имплементиране. Овдје се не мисли на програмску имплементацију, већ на у математичку, што ће имати за посљедицу и ефикаснију програмску имплементацију.

У циљу што ефектнијег рада функције и процедура: $\mathrm{Estimation}(\theta, Dom)$, $\mathrm{Sediment}(\theta, a)$, $\mathrm{CutOff}(\theta, \varepsilon)$ и $\mathrm{Minx}(D, \prec)$, одржава се структура ξ_k листи за сваку промјенљиву одлучивања x_k, $k \in \{1, \ldots, l\}$. Елементи ξ_k листи су уређени парови $\big(a, f_k(a)\big)$, за свако $a \in D_k$, $k \in \{1, \ldots, l\}$, који су још уређени у растућем поретку у односу на хеуристичку релацију $\prec_i$. Означимо ли са $n_k = |D_k|$, онда ξ_k се може представити као:

$$\xi_k = \langle \Big(a_{k,1}, f_k\big(a_{k,1}\big)\Big), , \ldots, \Big(a_{k,n_k}, f_k\big(a_{k,n_k}\big)\Big) \rangle, \quad k \in L, \tag{4.1.10}$$

водећи рачуна да су сљедећи услови испуњени:

$$(\forall k \in \{1, \ldots, l\})\ D_k = \{a_{k,1}, a_{k,2}, \ldots, a_{k,n_k}\}; \tag{4.1.11}$$

$$(\forall k \in \{1, \ldots, l\})(\forall i, j \in \{1, \ldots, n_k\})\ i \neq j \ \Leftrightarrow\ a_{k,i} \neq a_{k,j}; \tag{4.1.12}$$

$$(\forall k \in \{1, \dots, l\})(\forall i, j \in \{1, \dots, n_k\})\ \ i < j \ \Leftrightarrow\ a_{k,i} \prec_k a_{k,j}. \tag{4.1.13}$$

Ради лакшег означавања елемената ξ_k листи уводе се сљедеће ознаке:

$$\xi_k(i) = \big(a_{k,i}, f_k(a_{k,i})\big), \tag{4.1.14}$$

$$\xi_k(i,1) = a_{k,i},\ \ \xi_k(i,2) = f_k(a_{k,i}),\quad k \in L,\ i \in \{1, \dots, n_k\}. \tag{4.1.15}$$

На основу управо уведених појмова и ознака у формулама (4.1.10—15), SEDA функције и процедуре $\text{Estimation}(\theta, Dom)$ и $\text{Minx}(D, \prec)$ могу се израчунати на сљедећи начин:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \sum_{k=\theta}^{l} \begin{cases} \xi_k(1,2) & \text{if} \quad D_k \neq \emptyset, \\ +\infty & \text{if} \quad D_k = \emptyset. \end{cases} \tag{4.1.16}$$

$$\text{Minx}(D_k, \prec_k) = \xi_k(1,1). \tag{4.1.17}$$

За имплементацију функције $\text{CutOff}(\theta, \varepsilon)$ потребно је у ξ_k листама пронаћи уређене парове $\big(a, f_k(a)\big)$, који имају вриједност $f_k(a)$ мању од ограничења M_k, за сваку промјељиву одлучивања x_k, $k \in \{\theta + 1, \dots, l\}$. Подразумијева се да су M_k, дефинисани као у формули (4.1.7). Како су ξ_k листе већ уређене, бинарно претраживање је ефикасан алгоритам за проналажење тражених уређених парова. Уведемо ли μ оператор као:

$$\begin{aligned} &\mu y_{y<z} R(z) - \text{Најмање } y < z \text{ тако да важи } R(y), \\ &\qquad\qquad \text{ако } (\exists y \in \mathbb{N})\ y < z \Rightarrow R(y); \text{ иначе } z, \end{aligned} \tag{4.1.18}$$

тада се функција $\text{CutOff}(\theta, \varepsilon)$ имплементира на сљедећи начин:

$$M_k = \xi_k(1,2) + \varepsilon,\ \ k \in \{\theta + 1, \dots, l\}; \tag{4.1.19}$$

$$n'_k = \mu i_{i<n_k+1}[M_k \leq \xi_k(i,2)],\ \ k \in \{\theta + 1, \dots, l\}; \tag{4.1.20}$$

$$D'_k = \{\xi_k(i,1) \mid i \in \{1, \dots, n'_k - 1\}\},\ \ k \in \{\theta + 1, \dots, l\}. \tag{4.1.21}$$

Функција $\text{CutOff}(\theta, \varepsilon)$ враћа нове вриједности за ξ_k листе, које израчунавамо као:

$$\xi'_k = \langle\, \xi_k(i) \mid i \in \{1, \dots, n'_k\} \,\rangle,\ \ k \in \{\theta + 1, \dots, l\}. \tag{4.1.22}$$

Коначно нове вриједности за n'_k и ξ'_k, $k \in \{\theta + 1, \dots, l\}$ замјене старе, чиме се завршава имплементације функције $\text{CutOff}(\theta, \varepsilon)$:

$$n_k = n'_k \text{ и } \xi_k = \xi'_k, \qquad k \in \{\theta + 1, \dots, l\}. \tag{4.1.23}$$

Функција $\text{Sediment}(\theta, x)$, је имплементирана као секвенцијално претраживање кроз листе, ξ_k, $k \in \{\theta + 1, \dots, l\}$. Елементи, који задовољавају услове ограничења Φ и хеуристичке услове задржавају се у листама, остали се елиминишу. Ако дефинишемо предикат $\varphi(\theta, x, k, y)$ да буде тачан, ако и само ако, су све одређене вриједности промјењивих одлучивања x_1, x_2,…, $x_{\theta-1}$ и $x_\theta = x$, $x_k = y$ конзистентне са условима ограничења и хеуристичким условима, тада функцију можемо имплементирати као:

$$D'_k = \left\{ \xi_k(i,1) \mid i \in \{1, \dots, n_k\} \wedge \varphi\big(\theta, x, k, \xi_k(i,1)\big) \right\}, \quad k \in \{\theta + 1, \dots, l\}. \tag{4.1.24}$$

Као што је већ напоменуто нове вриједности за n_k и ξ_k, $k \in \{\theta + 1, \dots, l\}$ потребно је ажурирати у складу са D'_k, за будућа израчунавања и одржавања ових структура током рада SEDA-а.

На управо описан начин најважније помоћне функције SEDA-а своде се на операције са ξ_k листама. Комплексност ових операција не надмашује општу комплексност SEDA-а, тако се анализа њихове комплексности изоставља.

4.1.5. Опис SEDA-а

У овом подпоглављу, након увођења потребних функција и процедура, сљиједи опис SEDA алгоритма.

Алгоритам 4.1.1 *– Метод седиментације*

Након увођења функција и процедура потребних за рад SEDA, изложимо основни облик *методе седиментације*. Псеудокод SEDA-а је дат у псеудокоду 4.1.2. Из њега се јасно уочавају двије цјелине: главно тијело алгоритма, линије **(19)**—**(23)** и процедура $\text{FindSolution}(\theta, Dom)$, линије **(2)**—**(18)**. Улазни подаци су вриједности l, Dom, f, Ξ и F. Услови ограничења Φ, као и хеуристички услови, нијесу наведени као засебни улазни подаци, већ се подразумијева да су имплеметирани у функцији $\text{Sediment}(\theta, x)$, као што је то наведено у њеном псеудокоду 4.1.1.

У главном тијелу алгоритма, датом у виду функције прво се постављају почетне вриједности за промјењиве описане у потпоглављу 4.1.2, у линијама **(19)**—**(21)**, а потом се позива процедура $\text{FindSolution}(1, Dom)$, у линији **(22)**. На тај начин, ова рекурзивна процедура редом одређује вриједности

за промјењиве одлучивања, почевши од x_1, затим x_2, све док се не стигне до x_l. Након испитивања свих могућих вриједности, тј. свих допустивих рјешења проблема оптимизације, у складу условима ограничења Φ, као и хеуристичким условима, промјењива o ће садржати оптимално рјешење, уколико оно постоји. Промјењива $minimum$ вриједност функције критеријума оптималног рјешења $f(o)$, уколико оптимално рјешење o постоји. На крају рада главног тијела SEDA-а, ове двије промјењиве се враћају као резултати његовог рада, линија (**23**).

Псеудокод 4.1.2 *– Алгоритам методе седиментације*

```
1  SedimentationAlgorithm(l, Dom, f, Ξ, F)
2      procedure FindSolution(θ, Dom)
3          Dom' = Dom
4          D = D_θ
5          while D ≠ ∅ and Estimation(θ, Dom) < minimum do
6                  x_θ = Minx(D, ≺_θ)
7                  D = D \ {x_θ}
8                  Dom = Sediment(θ, x_θ)
9                  ε = minimum − Estimation(θ + 1, Dom)
10                 if ε > 0 then
11                     if θ = l ∨ F(θ) then ReportSolution()
12                                     else  Dom = CutOff(θ, ε)
13                                           FindSolution(θ + 1, Dom)
14                     endif
15                 endif
16                 Dom = Dom'
17         endwhile
18     end
19     o  = ⟨ ∅ | i = 1, … , l ⟩
20     x  = ⟨ ∅ | i = 1, … , l ⟩
21     minimum = +∞
22     FindSolution(1, Dom)
23     return ⟨o, minimum⟩
24 end
```

Ако оптимално рјешење не постоји, тј. не постоји ни једно допустиво рјешење, онда ће алгоритам вратити као резултат рада њихове почетне вриједности $\langle\langle\ \emptyset \mid i = 1, \dots, l\ \rangle, +\infty\rangle$. То омогућава лако препознавање да ли за дати проблем постоји оптимално рјешење.

Као што је већ раније наведено, процедура $\text{FindSolution}(\theta, Dom)$, рекурзивно испитује простор рјешења. Ради једноставнијег објашњења како функционише алгоритам, нека су одређене вриједности за промјењиве одлучивања $\{x_1, \dots, x_{\theta-1}\}$. Један од улазних података је скуп домена

$\{D_1, \dots, D_l\}$. Подразумијевамо да је овај скуп конзистентан са избором вриједности $\{x_1, \dots, x_{\theta-1}\}$. То значи да сви елементи скупова домена $\{D_\theta, \dots, D_l\}$, не крше услов ограничења Φ, у односу на вриједности промјењивих одлучивања $\{x_1, \dots, x_{\theta-1}\}$, које су већ одређене. Сљедећа вриједност коју треба одредити је за промјењиву одлучивања x_θ.

На почетку рада процедуре FindSolution(θ, Dom), прво се у помоћној промјењивој Dom', сачува вриједност улазне промјењиве Dom, у којој су просљеђени домени промјењивих одлучивања, линија (**3**). Такође, посебно се чува у помоћној промјењивој D, вриједност домена D_θ за промјењиву одлучивања x_θ, линија (**4**). Уколико то буде потребно, SEDA треба да испита све елементе скупа D, као могуће вриједности за промјењиву одлучивања x_θ. Након испитивања неке вриједности $a \in D$, бришемо је из скупа D. На тај начин, алгоритам испитује све вриједности промјењиве одлучивања x_θ, док је $D \neq \emptyset$ и постоји шанса за побољшање тренутно најбољег рјешења тј., Estimation$(\theta, Dom) < minimum$, у ***while*** петљи (**5**)—(**17**).

У линији (**6**) бирамо минимални $\prec_\theta$ елемент скупа D за вриједност промјењиве одлучивања x_θ. Након одабира тај елемент уклањамо из скупа D, у линији (**7**). Избор вриједност за промјењиву одлучивања x_θ утиче на елементе скупова домена промјењивих одлучивања $\{D_{\theta+1}, \dots, D_l\}$. Неки од ових елемената нијесу у складу условима ограничења Φ, за ново изабрану вриједности промјењиве x_θ. Такође, постоји могућност да хеуристике које се користе омогућавају одбацивање појединих елемената, без бојазни да ће на тај начин у простору рјешења бити изгубљено оптимално рјешење. Одбацивање, као што је то већ описано у потпоглављу 4.1.3, се врши позиварем функције Sediment(θ, x_θ), која враћа нове скупове домена промјењивих одлучивања, у линији (**8**). Напоменимо, још једном да ће вриједности домена бити промијењене само за промјењивих одлучивања чија вриједност још није одређена.

Након одређивања вриједности за промјењиву одлучивања x_θ и ажурирања скупа домена $\{D_{\theta+1}, \dots, D_l\}$, процјена функције критеријума f може бити различита у односу на ону процјену направљену у линији (**5**). Умјесто директног поређења Estimation$(\theta + 1, Dom) < minimum$, рачунамо разлику $minimum -$ Estimation$(\theta + 1, Dom)$ и смјештамо у помоћну промјењиву ε,

линија (**9**). Затим се у линији (**10**) провјерава да ли је $\varepsilon > 0$? Уколико је овај услов испуњен, алгоритам може наставити са конструкцијом допустивих рјешења у линијама (**11**)—(**14**). Ако услов $\varepsilon > 0$ није задовољен, онда посљедња додјела вриједности промјенљиве одлучивања не води ка побољшању тренутно најбољег рјешења, па треба наставити са сљедећом расположивом вриједношћу из скупа D, док год је он непразан.

У линији (**11**) испитујемо да ли је $\theta = l$ или $F(\theta)$? У случају да јесте конструисано је ново допустиво рјешење проблема, па сходно томе треба провјерити да ли је то рјешење боље од тренутно најбољег позивом процедуре ReportSolution().

Ако услови $\theta = l$ и $F(\theta)$ нијесу испуњени, тада је $\theta < l$, па треба наставити са конструкцијом допустивих рјешења. Прије него ли се конструкција настави, из скупа домена неодређених промјенљивих одлучивања, треба елиминисати оне елементе, који имају превелику вриједност функције критеријума, позивањем функције CutOff(θ, ε), линија (**12**). Након одређивања нових вриједности домена, може се наставити са конструкцијом допустивих рјешења рекурзивним позивом функције FindSolution$(\theta + 1, Dom)$, линија (**13**).

На крају, враћају се старе вриједности скупа домена промјенљивих одлучивања у линији (**16**), како би се наставило са испитивањем осталих могућих вриједности за промјенљиву x_θ.

Backtracking механизам SEDA је *„сакривен“* у рекурзивној формулацији процедуре FindSolution(θ, Dom). Нерекурзивна формулација је комплекснија, али и ефикаснија са становишта употребе рачунарских ресурса. Управо, ради лакшег излагања алгоритма изабрана је рекурзивна варијанта. Сви тест примјери, који су описани у наставку, рјешавани су нерекурзивном верзијом алгоритма. Претварање рекурзивне формулације алгоритма у нерекурзивну је, мање или више, техничка ствар па је изостављамо.

4.1.6. Основна својства SEDA-а

У наставку овог подпоглавља биће приказана основна својстава SEDA-а. Прво ће бити доказана тотална коректност алгоритма, а потом се даје крађи осврт на питање комплексности алгоритма.

Тотална коректност алгоритма

SEDA врши елиминацију елемената домена промјењивих одлучивања у линијама (**5**), (**8**), (**10**) и (**12**) његовог псеудокода. То су линије које суштински стварају разликују између SEDA и GBBA. Будући да је тотална коректност GBBA већ доказана у теореми 2.1.3, за доказ коректности SEDA-а биће довољно доказати да елиминације уведене у линијама (**5**), (**8**), (**10**) и (**12**) не спрјечавају SEDA да пронађе тачно оптимално рјешење проблема. У наставку текста овог подпоглавља реферисање на линије кода подразумијевају да се ради о псеудокоду 4.1.2, који описује SEDA. Такође, промјенљиве које се помињу у наставку подпоглавља су из истог псеудокода, сем уколико нијесу дефинисане у потпоглављима 4.1.1—3.

Лема 4.1.1 *– Лема о корацима (5) и (10)*

Ако су вриједности промјењивих одлучивања $\{x_1, \dots, x_{\theta-1}\}$ одређене у допустивом рјешењу x, тада је допустиво рјешење x није боље од тренутно најбољег рјешења o, ако је:

$$\text{Estimation}(\theta, Dom) \geq minimum = f(o). \tag{4.1.25}$$

Доказ.

Ако су вриједности промјењивих одлучивања $\{x_1, \dots, x_{\theta-1}\}$ одређене, тада се функција $\text{Estimation}(\theta, Dom)$ израчунава, сходно формули (4.1.6), као:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \sum_{k=\theta}^{l} \begin{cases} \min\limits_{y \in D_k} f_k(y) & : \quad D_k \neq \emptyset, \\ +\infty & : \quad D_k = \emptyset. \end{cases} \tag{4.1.26}$$

Ако је домен D_k, празан скуп, за неко $k \in \{\theta, \dots, l\}$, тада је, на основу формуле (4.1.26), $\text{Estimation}(\theta, Dom) = +\infty$, па неједнакост $\text{Estimation}(\theta, Dom) \geq minimum$, тривијално важи.

Ако су домени D_k, непразни скупови, за свако $k \in \{\theta, \dots, l\}$, тада је, на основу формуле (4.1.26):

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \sum_{k=\theta}^{l} \min_{y \in D_k} f_k(y). \tag{4.1.27}$$

Прва сума у формули (4.1.27) је константна, будући да су вриједности промјењивих одлучивања $\{x_1, \dots, x_{\theta-1}\}$ одређене. Друга сума у (4.1.27) процјењује вриједност функције критеријума за промјењиве одлучивања, које нијесу одређене $\{x_\theta, \dots, x_l\}$, тако што узима њихову минималну вриједност на домену: $\min_{y \in D_k} f_k(y)$. Ова процјена не мора нужно да одговара допустивом рјешењу, али је свакако минимална. Због тога, свако допустиво рјешење има вриједност функције критеријума које је веће или једнако од $\text{Estimation}(\theta, Dom)$. Зато, ако важи $\text{Estimation}(\theta, Dom) \geq minimum$, за одређене промјењиве одлучивања $\{x_1, \dots, x_{\theta-1}\}$, тада не може постојати допустиво рјешење x, које садржи одређене промјењиве одлучивања $\{x_1, \dots, x_{\theta-1}\}$, а да је при томе боље од тренутно најбољег рјешења o. **QED**.

Посљедица 4.1.1 *– Кораци (5) и (10) су коректни*

Елиминација елемената домена неодређених промјењивих одлучивања $\{x_{\theta+1}, \dots, x_l\}$ у линијама **(5)** и **(10)** псеудокода SEDA-а, не спријечава SEDA да пронађе оптимално рјешење проблема оптимизације.

Елиминација елемената домена у линији (8) псеудокода SEDA-а врши се за вријеме позивања функције $\text{Sediment}(\theta, x)$. Као што је то већ описану у потпоглављу 4.1.3, елиминишу се они елементи који не задовољавају услове ограничења Φ или хеуристиче услове. Услови ограничења Φ, сами по себи, не могу спријечити SEDA да пронађе оптимално рјешење, јер они одређују која рјешења сматрамо допустивим. За хеуристичке услове узима се као претпоставка да су такви, да не могу елиминацијом елемената домена спријечити SEDA да пронађе оптимално рјешење. Стога је потребно у свакој конкретној примјени доказати да хеуристички услови не нарушавају комплетну претрагу простора рјешења.

На основу горњих чињеница и претпоставки формулишемо сљедећу посљедицу, без њеног доказа.

Посљедица 4.1.2 *– Корак (8) је коректнан*

Елиминација елемената домена неодређених промјењивих одлучивања $\{x_{\theta+1}, \dots, x_l\}$ у линији **(8)** псеудокода SEDA-а, не спријечава SEDA да пронађе оптимално рјешење проблема оптимизације.

***Лема 4.1.2** – Лема о кораку (12)*

Ако постоји оптимално рјешење x различито од тренутно најбољег рјешења o и при томе вриједности промјењљивих одлучивања $\{x_1, \dots, x_\theta\}$ су одређене, тада ако означимо са:

$$\varepsilon = minimum - \text{Estimation}(\theta + 1, Dom), \tag{4.1.28}$$

важи еквиваленција:

$$\varepsilon > 0 \iff (\forall k \in \{\theta + 1, \dots, l\})\ f_k(x_k) \leq \min_{y \in D_k} f_k(y) + \varepsilon. \tag{4.1.29}$$

Ако искористимо нотацију из описа функције $\text{CutOff}(\theta, \varepsilon)$ из потпоглавља 4.1.3, тада се еквиваленција (4.1.29) краће записује као:

$$\varepsilon > 0 \iff (\forall k \in \{\theta + 1, \dots, l\})\ x_k \in D'_k. \tag{4.1.30}$$

Прије него ли се докаже лема 4.1.2, примјетимо да пошто су првих θ промјењљивих одлучивања оптималног рјешења x одређене, мора да важи:

$$(\forall k \in \{\theta, \dots, l\})\ D_k \neq \emptyset. \tag{4.1.31}$$

Доказ.

Доказује се прво $(\Leftarrow)$ смјер. На основу дефиниције промјењљиве $minimum$, у потпоглављу 4.1.2, важи: $minimum = f(o_1, \dots, o_l)$. Ако постоји оптимално рјешење x, различито од o, тада је $f(o_1, \dots, o_l) > f(x_1, \dots, x_l)$. Из тога непосредно слиједи:

$$minimum - f(x_1, \dots, x_l) > 0 \quad \Leftrightarrow \quad minimum > f(x_1, \dots, x_l). \tag{4.1.32}$$

На основу претпоставки о функцији $\text{Estimation}(\theta + 1, Dom)$, наведеним у потпоглаљу 4.1.3 и претпоставке $(\forall k \in \{\theta + 1, \dots, l\})\ x_k \in D'_k$ важи неједначина:

$$f(x_1, \dots, x_l) \geq \text{Estimation}(\theta + 1, Dom). \tag{4.1.33}$$

Из неједначина (4.1.32) и (4.1.33) непосредно изводимо $(\Leftarrow)$ смјер еквиваленције, дате у формули (4.1.29):

$$minimum - \text{Estimation}(\theta + 1, Dom) > 0 \iff \varepsilon > 0. \tag{4.1.34}$$

Други смјер еквиваленције $(\Rightarrow)$, дате у формули (4.1.29), доказује се свођењем на апсурд. Уведимо ознаке:

$$N_k = \min_{y \in D_k} f_k(y), \quad k \in \{\theta + 1, \dots, l\}; \tag{4.1.35}$$

$$N'_k = \min_{y \in D'_k} f_k(y), \quad k \in \{\theta + 1, \dots, l\}. \tag{4.1.36}$$

Из описа SEDA датог у потпоглављима 4.1.3—5, непосредно се закључује да важи $(\forall k \in \{\theta + 1, \dots, l\})\ D'_k \subseteq D_k$, пошто функције $\text{Sediment}(\theta, e_\theta)$ и $\text{CutOff}(\theta, \varepsilon)$ само елиминишу елементе из скупа домена. Због тога важи:

$$(\forall k \in \{\theta + 1, \dots, l\})\ N'_k \geq N_k. \tag{4.1.37}$$

Претпоставимо да важи $\varepsilon > 0$ и $(\exists k \in \{\theta + 1, \dots, l\})\ x_k \notin D'_k$. Нека је $s \in \{\theta + 1, \dots, l\}$, такво да важи:

$$x_s \notin D'_s \iff f_s(x_s) \geq \min_{y \in D'_s} f_s(y) + \varepsilon \iff f_s(x_s) \geq N_s + \varepsilon. \tag{4.1.38}$$

Пошто је $minimum > f(x_1, \dots, x_l)$ и имајући у виду својства низова N_k и N'_k, $k \in \{\theta + 1, \dots, l\}$, исказаним у формулама (4.1.37) и (4.1.38), важи сљедећи низ неједначина и једначина:

$$\begin{aligned}
&minimum > f(x_1, \dots, x_l) = \\
&= \sum_{k=1}^{l} f_k(x_k) = \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{l} f_k(x_k) = \\
&= \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{s-1} f_k(x_k) + f_s(x_s) + \sum_{k=s+1}^{l} f_k(x_k) \geq \\
&\geq \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{s-1} N'_k + (N'_s + \varepsilon) + \sum_{k=s+1}^{l} N'_k = \\
&= \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{l} N'_k + \varepsilon \geq \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{l} N_k + \varepsilon = \\
&= \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{l} \min_{y \in D_k} f_k(y) + \varepsilon \\
&= \text{Estimation}(\theta + 1, Dom) + \varepsilon = \\
&= \text{Estimation}(\theta + 1, Dom) + \\
&\qquad + minimum - \text{Estimation}(\theta + 1, Dom) = \\
&= minimum.
\end{aligned} \tag{4.1.39}$$

Повежемо ли почетак и крај горњег низа неједначина и једначина добијамо да је $minimum > minimum$, што је контрадикција. Тиме је закључен доказ ове леме. **QED**.

Посљедица 4.1.3 *– Корак (12) је коректан*

Примјена CutOff(θ, ε) функције у линији **(12)** не спрјечава SEDA да пронађе оптимално рјешење проблема оптимизације.

Доказ.

Претпоставимо да постоји оптимално рјешење x, различито од тренутно најбољег рјешења o, у неком од корака рада SEDA-а. Ако се одређене вриједности промјенљивих одлучивања разликују од одговарајућих у оптималном рјешењу x, тада било каква елиминација елемената из скупа домена неће спријечити SEDA, у проналажењу оптималног рјешења, јер то у том случају није ни могуће.

Уколико су одређене вриједности промјенљивих одлучивања $\{x_1, \dots, x_\theta\}$, идентичне са оним у оптималном рјешењу x, тада лема 4.1.2 осигурава да примјена CutOff(θ, ε) функције, у линији **(12)** псеудокода SEDA-а, не спрјечава SEDA да пронађе оптимално рјешење x. **QED**.

Теорема 4.1.1 *– Теорема о потпуној коректности SEDA*

SEDA је тотално коректан алгоритам.

Доказ.

Као што је већ раније наведено, SEDA врши редукује скупове домена промјенљивих одлучивања у линијама: **(5)**, **(8)**, **(10)** и **(12)**, псеудокода СЕДА. Без ових редукције алгоритам би се понашао исто као и BFA, или GBBA, за које је већ доказано да су тотално коректни у теоремиама 2.1.1 и 2.1.3. Стога је довољно доказати да редукције у наведеним корацима псеудокода не спрјечава SEDA да пронађе оптимално рјешење.

Посљедица ***4.1.1*** доказује да елиминације у корацима **(5)** и **(10)**, псеудокода не спрјечава SEDA да пронађе оптимално рјешење, посљедица 4.1.2 доказује исто за корак **(8)** и коначно посљедица 4.1.3 обезбјеђује исти доказ за корак **(12)**. Из свега закључујемо да је SEDA је тотално коректан алгоритам. **QED**.

4.1.7. О комплексности SEDA-а

Прецизнија оцјена комплексност SEDA-а, сем очигледне чињенице да се ради о експоненцијалној комплексности, није могућа. У општем случају тешко је предвидјети могуће редукције домена, које настају у корацима **(5)**, **(8)**, **(10)** и **(12)** његовог псеудокода, од којих зависи комплексност SEDA-а.

Да би илустровали ову тврдњу претпоставимо да су сви домени исте кардиналности d. Уколико нема редукција SEDA понаша једнако као и BFA, који има комплексност $O(d^l)$, како смо то већ утврђено у теореми 2.1.2. Ако се скупови домена редукују за по један елемент у сваком кораку рада SEDA-а, тада је његова комплексност $O\big(d(d-1)(d-2)\cdots(d-l+1)\big) = O\left(\frac{d!}{l!}\right)$, под условом да $d > l$. Коначно, претпоставимо да се кардиналност домена редукује за коефицијент $p \in (0,1)$, у сваком кораку рада SEDA-а. Комплексност у том случају можемо процијенити као $O\Big(d \cdot (pd) \cdot (p^2 d) \cdots (p^{l-1} d)\Big) = O\left(d^l p^{\frac{l(l-1)}{2}}\right)$.

Ова три једноставна примјера показују колико је у општем случају тешко одредити комплексност SEDA-а. Када се алгоритам примјењује на конкретном примјеру, лакше је утврдити комплексност, што ће и бити случај, када будемо изложили примјене овог алгоритма.

4.2. АЛГОРИТАМ СЕДИМЕНТАЦИЈЕ + ХЕУРИСТИКА „ПРОЦЈЕНИ И ПРЕУРЕДИ“

Ефикасност рада SEDA-а веома зависи од поретка којим се испитују вриједности за промјењиве одлучивања. Вриједност функције критеријума одређујемо као:

$$f(x_1, \dots, x_l) = \sum_{k=1}^{l} f_k(x_k). \tag{4.2.1}$$

Добар поредак је онај, у којем се прво одређују вриједност за промјењиве одлучивања чије су вриједности $f_k(x_k)$, веће. Другим ријечима под *„добрим“* поретком би подразумијевали пермутацију $\omega: L \to L$, такву да за оптимално рјешење x, важи:

$$f_{\omega(1)}\big(x_{\omega(1)}\big) \geq f_{\omega(2)}\big(x_{\omega(2)}\big) \geq \cdots \geq f_{\omega(l)}\big(x_{\omega(l)}\big). \tag{4.2.2}$$

Пермутација које задовољавају услов (4.2.2) има више, међутим нијесу све међу њима једнако добре. У просјеку, ако користимо *„добре“* пермутације значајно се скраћује рад SEDA-а. Из примјера се показује да могу постојати и боље пермутације од њих, али у општем случају, као што је већ наведено, коришћењем пермутација које задовољавају услов (4.2.2) скраћујемо рад SEDA-а.

Да би примијенили ову технику, треба одговорити на питање како одредити *„добру“* пермутацију ω, по којој ће се вршити одређивање вриједности промјењивих одлучивања. Оптимално рјешење није унапријед познато, да јесте не би постојао мотив да конструишемо алгоритам, који га проналази. Проблем се рјешава тако, што пустимо ограничен број пута SEDA да ради. При томе, сваки покушај је временски ограничен и сваки покушај ради за различити поредак (пермутацију) промјењивих одлучивања. Најбоље рјешење које добијемо на тај начин, основа је за стварање поретка ω. Затим, покренемо SEDA да ради по ω поретку, без ограничења времена, дозвољавајући да се претражи комплетан простор рјешења. На крају добијамо оптимално рјешење, уколико оно постоји. Описани поступак/хеуристику називамо *хеуристиком процјене и прегруписавања*, или на енглеском језику *Estimation & Rearrangement Heuristic* (ERH). Примјеном ERH-е прије SEDA-а, добијамо нови алгоритам за егзактно рјешавање комбинаторних проблема. Називамо га *методом седиментације са хеуристиком процјене и прегруписавања*, или на енглеском језику *Sedimentation Algorithm with an Estimation & Rearrangement Heuristic* (SEDA+ERH). Убудуће, скраћеница SEDA+ERH се користи за реферисање на овај алгоритам.

Опишимо надаље детаљније SEDA+ERH. Увођењем поретка ω, којим се утврђује редослијед одређивања промјењивих одлучивања мијењају се формулација SEDA-а. На примјер формула функције критеријума мијења се у:

$$f(x_1, \dots, x_l) = \sum_{k=1}^{l} f_{\omega(k)}\left(x_{\omega(k)}\right). \tag{4.2.3}$$

Потребне измјене су једноставне, па сматрамо да није потребно изнова формулисати SEDA у случају да користи поредак ω. Умјесто тога, обратићемо посебну пажњу на измјене које су потребне у његовом псеудокоду 4.1.2.

Алгоритам 4.2.1 –

Метод седиментације прилагођен за хеуристику „процијени и преуреди“

На свим мјестима у псеудокоду SEDA-а, гдје се приступа θ промјенљивој одлучивања треба замијенити са приступом $\omega(\theta)$ промјенљивој одлучивања. Такође, бришемо линије кода **(19)** “$o = \langle \emptyset \mid i = 1, \dots, l \rangle$” и **(21)** “$minimum = +\infty$”. Промјенљиве o и $minimum$ биће глобалне промјенљиве SEDA+ERH. Алгоритам SEDA, који је преправљен да ради заједно са ERH називамо SedimentationAlgorithm$\Omega(l, Dom, f, \Xi, F, \omega)$ и дат је у псеудокоду 4.2.1.

***Псеудокод 4.2.1** – Метод седиментације прилагођен за ERH*

```
1  SedimentationAlgorithmΩ(l, Dom, f, Ξ, F, ω)
2      procedure FindSolution(θ, Dom)
3          Dom′ = Dom
4          D = D_ω(θ)
5          while D ≠ ∅ and Estimation(θ, Dom) < minimum do
6              x_ω(θ) = Minx(D, ≺_ω(θ))
7              D = D \ {x_θ}
8              Dom = Sediment(ω(θ), x_ω(θ))
9              ε = minimum − Estimation(θ + 1, Dom)
10             if ε > 0 then
11                 if θ = l ∨ F(θ) then ReportSolution()
12                                 else  Dom = CutOff(θ, ε)
13                                       FindSolution(θ + 1, Dom)
14                 endif
15             endif
16             Dom = Dom′
17         endwhile
18     end
19     FindSolution(1, Dom)
20     return ⟨o, minimum⟩
21 end
```

За формулацију SEDA+ERH потребне су нам сљедеће функције и процедуре:

1. RunForLimitedTime$(function, time)$ – функција која покреће израчунавање функције $function$ у трајању од највише $time$ секунди рада.

Ова функција враћа уређен пар $\langle result, finished \rangle$, гдје $result$ представља вриједност функције, уколико је она израчуната у временском ограничењу $time$, а $finished$ је *flag*, који је тачан, ако је израчунавање завршено у временском ограничењу, односно нетачан уколико није.

2. RandomPermutation(ω) – функције која враћа случајну пермутацију скупа ω.
3. SortAccordingTo(C) – функција која за улазни низ $C = \{c_1, \dots, c_l\}$, израчунава и враћа низ ω, такав да:

$$C_{\omega(1)} \geq C_{\omega(2)} \geq \cdots \geq C_{\omega(l)}. \tag{4.2.4}$$

Псеудокод 4.2.2 *– Алгоритам таложења са хеуристиком процјене и прегруписавања*

```
1  SEDA_ERH(l, Dom, f, Ξ, F, SEDArestarts, EstT)
2      minimum = +∞
3      o = ⟨ ∅ | i = 1, … l ⟩
4      ω = {1, … , l}
5      for i = 1 to SEDArestarts do
6          ⟨⟨o′, minimum′⟩, completed⟩ =
7              RunForLimitedTime(SedimentationAlgorithmΩ(l, Dom, Ξ, f, ω), EstT)
8          if completed then return ⟨o′, minimum′⟩ endif
9          ω = RandomPermutation(ω)
10     endfor
11     if minimum < +∞ then
12         C = ⟨ f_k(o_k) | k ∈ {1, … , l}⟩
13         ω = SortAccordingTo(C)
14     else
15         ω = {1, … , l}
16     endif
17     return SedimentationAlgorithmΩ(l, Dom, f, Ξ, F, ω)
18 End
```

Алгоритам 4.2.2 –

Метод седиментације и хеуристика „процјени и прегледи“

Алгоритам SEDA+ERH дат је у виду псеудокода 4.2.2. Садржи, поред улазних промјенљивих: l, Dom, Ξ и f, који су потребни за SEDA, још и улазне промјенљиве: $SEDArestarts$ и $EstT$. Улазна промјенљива $SEDArestarts$ одређује број процјена, тј. колико пута се дозвољава да ограничено ради SEDA, док $EstT$ одређује трајање ограниченог извршавања SEDA-а.

Линије **(2)**—**(4)** постављају почетне вриједности за глобалне промјенљиве. Даље, у линијама **(5)**—**(10)** врше се процјене оптималног рјешења, тако што

се временски ограничено позива SEDA-а. Уколико се у току ових покушаја дође до оптималног рјешења, тако што покренемо SEDA, који заврши рад прије временског ограничења $EstT$, тада се може прекинути рад и вратити вриједност $\langle o', minimum' \rangle$, линија (**8**). Након завршетка временски ограничених процјена испитује се услов $minimum < +\infty$ у линији (11). Ако је он испуњен, онда се током утврђивања процјена дошло до бар једног допустивог рјешења. Најчешће се током процјена дође до више допустивих рјешења, од којих најбоље достигнуто се налази у промјењливој o, а вриједност његове функције критеријума је смјештена у промјењливој $minimum$. Дакле, уколико је $minimum < +\infty$ ствара се поредак ω, сходно вриједностима функције критеријума за вриједности функције критеријума, линије (**12**)—(**13**). Уколико услов $minimum < +\infty$ није испуњен, то значи да у току процјена није пронађено допустиво рјешење. У том случају за поредак ω, узимамо најједноставнији $\omega = \{1, \dots, l\}$, у линији (**15**). Коначно позивамо на неограничено вријеме SEDA у поретку ω, и резултате његовог израчунавања сматрамо за коначне, линија (**18**).

4.3. АЛГОРИТАМ СЕДИМЕНТАЦИЈЕ + ХЕУРИСТИКА „ПРОЦЈЕНИ И ПРЕУРЕДИ“+ СТРАТЕГИЈА „ПОДИЈЕЛИ, ПА ВЛАДАЈ“

Метод седиментације, могуће је даље убрзати примјеном стратегије *„подијели, па владај“,* на енглеском језику та се стратегија оптимизације назива *Divide & Conquer Strategy,* даље у тексту D&C. Њу кратко можемо описати као покушај да се проблем разбије на подпроблеме, они се ријеше и на основу тих рјешења изградимо рјешење полазног проблема. Ову стратегију није могуће увијек примијенити, како иначе, тако и у општем случају за SEDA. Наставку текста описаћемо услове за примјену D&C стратегије у оквиру SEDA-а.

4.3.1. Опис D&C стратегије

Дефиниција 4.3.1 *– Скуп свих могућих вриједности промјен. одлучивања*

Користећи ознаке из потпоглавља 2.1.1, за скуп домена $Dom = \{D_1, \dots, D_l\}$, промјењливих одлучивања $X = \{x_1, \dots, x_l\}$, дефинишемо *универзални скуп*, у ознаци A, као:

$$A = \bigcup_{i=1}^{l} D_i = \cup Dom. \tag{4.3.1}$$

Дефиниција 4.3.2 *– Релација неконфликтности, релација конфликтности за чланове универзалног скупа*

Симетричну нерефлексивну бинарну релацију $\rho \subseteq A^2$, називамо *релацијом неконфликтности*, уколико за свако допустиво рјешење $x = (x_1, \dots, x_l)$, проблема оптимизације важи да су $x_i \ \rho \ x_j$, за $i \neq j$, и $i, j \in L$. Својство неконфликтности релације $\rho \subseteq A^2$ можемо исказати сљедећом формулом:

$$\left(\forall x \in \prod_{k=1}^{l} D_k\right) \Phi(x) \ \Leftrightarrow \ \left((\forall i, j \in L) \ i \neq j \ \Rightarrow \ x_i \ \rho \ x_j\right). \tag{4.3.2}$$

Симетричну, рефлексивну релацију *конфликтности*, у ознаци $\bar{\rho} \subseteq A^2$, дефинишемо као релацију супротну релацији неконфликтности.

Избор релација неконфликтности и конфликтности није једноставан. Штовише, за неке проблеме оптимизације није могуће изабрати такве релација. Са друге стране, за неке друге проблеме оптимизације је могуће изабрати такве релације, које значајно редукују вријеме рада SEDA-а и SEDA+ERH. Питању избора посвећујемо пажњу, код конкретних примјера у наставку текста.

Проширимо даље дефиницију релација неконфликтности и конфликтности на подскупове скупа индекса L.

Дефиниција 4.3.3 *– Релација неконфликтности, за подскупове скупа индекса*

За валуацију $x \in \prod_{k=1}^{l} D_k$, и за било која два подскупа $\omega_1, \omega_2 \subseteq L$, кажемо да су *неконфликтни у валуацији* x, ако и само ако $x_i \ \rho \ x_j$ за свако $i \in \omega_1$ и $j \in \omega_2$. Релацију неконфликтности у валуацији x, означавамо са ρ_x и можемо је исказати формулом:

$$\omega_1 \ \rho_x \ \omega_2 \ \Leftrightarrow \ \ (\forall i \in \omega_1)(\forall j \in \omega_2) \ x_i \ \rho \ x_j. \tag{4.3.3}$$

Дефиниција 4.3.4 *– Релација конфликтности, за подскупове скупа индекса*

За валуацију $x \in \prod_{k=1}^{l} D_k$, и за било која два подскупа $\omega_1, \omega_2 \subseteq L$, кажемо да су *конфликтни у валуацији* x, ако и само ако постоји $i \in \omega_1$ и постоји $j \in \omega_2$,

тако да $x_i \bar{\rho} x_j$. Релацију конфликтности у валуацији x, означавамо са $\bar{\rho}_x$ и можемо је исказати формулом:

$$\omega_1 \bar{\rho}_x \omega_2 \Leftrightarrow (\exists i \in \omega_1)(\exists j \in \omega_2)\, x_i \bar{\rho} x_j. \tag{4.3.4}$$

Дефиниција 4.3.5 *– Рефлексивно транзитивно симетрично затворење*

За бинарну релацију $\sigma \subseteq L^2$, њено *рефлексиво транзитивно симетрично затворење* означавамо са $\sigma^{\equiv}$. Очигледно релација $\sigma^{\equiv}$ је релација еквиваленције.

Дефиниција 4.3.6 *– Класа еквиваленције*

За релацију еквиваленције $\sigma \subseteq L^2$ и елемент $i \in L$ означавамо са $[i]_\sigma$ *класу еквиваленције* и дефинишемо као скуп $[i]_\sigma = \{j \in L \mid i\ \sigma\ j\}$.

Дефиниција 4.3.7 *– Скуп свих класа еквиваленције*

За релацију еквиваленције $\sigma \subseteq L^2$ означавамо са $L_{/\sigma}$ *скуп свих класа еквиваленције*.

Дефиниција 4.3.8 *– Минимална валуација*

Валуацију $x_0 \in \prod_{k=1}^{l} D_k$ зовемо *минималном валуацијом* ако важи услов:

$$\left(\forall y \in \prod_{k=1}^{l} D_k\right) f(x_0) \leq f(y). \tag{4.3.5}$$

Примјетимо, минимална валуација не мора да буде јединствена. Такође, најчешће минималне валуације нијесу допустива рјешења, што чини њихову конструкцију једноставном. Оне су допустива рјешења само у тривијалним инстанцама проблема оптимизације.

Да би формулисали D&C стратегију потребне су нам сљедеће функције и процедуре:

1. $\text{GetMinValuation}(l, Dom, f)$ – функција која проналази и враћа минималну валуацију.
2. $\text{Solver}(l, Dom, \Xi, f, \omega)$ – функција, која може да ријеши ПО. Враћа вриједност $\langle o, minimum \rangle$, гдје o, представља оптимално рјешење, а $minimum$ вриједност функције критеријума оптималног рјешења.

Овдје ћемо у првом реду подразумијевати да се ради о некој од верзија SEDA-а или SEDA+ERH мада је сама D&C стратегија, овдје формулисана, тако да буде независна од било којег конкретног приступа рјешавању проблема оптимизације.

Алгоритам 4.3.1 – Стратегија „подијели, па владај“

Алгоритам D&C дат је у виду псеудокода 4.3.1. Његове улазне промјењиве су: l, Dom, Ξ и f. Подразумијевамо да је скуп услова, који одређује које су валуације допустива рјешења, имплеметирана у $\text{Solver}(l, Dom, f, F, \Xi, \omega)$ функцији. Алгоритам рјешава подпроблеме полазног проблема, све док њихова рјешења не постану неконфликтна, при томе неке ће подпроблеме обједињавати, јер имају међусобно конфликтна рјешења. Подпроблеме представљамо као подскупове скупа индекса промјењивих одлучивања L.

***Псеудокод 4.3.1** – Алгоритам стратегије „подјели и владај“ за рјешавање проблема оптимизације*

```
1  D&C(l, Dom, Ξ, f)
2      x = GetMinValuation(l, Dom, Ξ, f);   ≈ = ρ̄_x^≡
3      U = L_/≈;   S = ∅
4      until U ≠ ∅ do
5          ω ∈ U
6          ⟨x_ω, minimum_ω⟩ = Solver(l, Dom, f, Ξ, F)
7          if minimum_ω = +∞ then return ⟨∅, +∞⟩
8          x = x_|L\ω ∪ x_ω;  ≈ = ρ̄_x^≡
9          Ω = {φ ∈ U ∪ S | ω ≈ φ}
10         if Ω = {ω} then
11             S = S ∪ {ω};   U = U\{ω}
12         else
13             S = S\Ω
14             U = U\Ω;  U = U ∪ {∪Ω}
15         endif
16     enduntil
17     return ⟨x, f(x)⟩
18 end
```

На почетку, у линији **(2)**, узимамо да је x, било која минимална валуација. Она најчешће није допустиву рјешење. Да би то била требамо у даљем раду уклонити из ње конфликтне додјеле вриједности промјењивих одлучивања. У наставку рада *„поправљамо“* валуацију x, све док она не буде допустиво рјешење. Испоставиће се да када алгоритам стане са радом, рјешење које добијемо биће не само допустиво, већ и оптимално. У том циљу, такође

у линији (**2**), дефинишемо и релацију $\approx$, као рефлексивно транзитивно симетрично затворење релације конфликтних индекса промјењивих одлучивања, тј. $\bar{\rho}_x^{\equiv}$.

Даље, дефинишемо два скупа U и S, линија (**3**). Скуп U садржи неријешене подпроблеме, док скуп S, садржи ријешене подпроблеме. Почетна вриједност скупа U је скуп класа еквиваленције, када скуп индекса промјењивих одлучивања L пресјечемо релацијом еквиваленције $\approx$, тј. $U = L_{/\approx}$. Почетна вриједност скупа S је празан скуп.

Главна идеја D&C стратегије, овдје примијењене, је да рјешавамо подпроблеме проблема оптимизације за свако $\omega \in U$. Уколико рјешење подпроблема ω није конфликтно са осталим подпроблемима у U и S, пребаци се у скуп S. Ако је рјешење конфликтно, онда се уклоне из U и S сви чланови (подпроблеме) који су у конфликту са ω, затим направимо њихову унију и прикључимо је скупу U, те даље наставимо да рјешавамо. Овај поступак се понавља, све док је скуп U непразан. На почетку рада скуп S је празан скуп, а на крају рада скуп U је празан скуп.

У линији (**4**), улази се у ***until*** петљу, коју извршавамо све док је $U \neq \emptyset$. Ако се ушло у петљу, прво бирамо било који елемент $\omega \in U$, линија (5), а затим се ријеши подпроблем ω и рјешење смјести у помоћну промјењиву x_ω, а вриједност функције критеријума у помоћну промјењиву $minimum_\omega$, линија (**6**). Ако је $minimum_\omega = +\infty$, то значи да подпроблем ω, нема рјешења. У том случају, у линији (**7**), се прекида рад алгоритма и враћа вриједности $\langle \emptyset, +\infty \rangle$, као показатељ да полазни ПО нема рјешења.

Ако подпроблем ω има рјешења, ажурира се валуацију x, тако што се у њој замијене рјешења за подпроблем ω, док остале вриједности $L \backslash \omega$ у валуацији x, остављамо непромијењеним. За нову вриједност валуације x, потребно је поново одредити релацију $\approx$, рефлексивно транзитивно симетрично затворење, релације конфликтних индекса промјењивих одлучивања, тј. $\bar{\rho}_x^{\equiv}$. Наведене операције у овом пасусу, налазе се у (**8**) линији псеудокода.

У линији (**9**) псеудокода одређујемо скуп Ω, као скуп свих оних подпроблема из скупова U и S, чије је рјешење у конфликту са рјешењем подпро-

блема ω. Скуп Ω ће увијек бити непразан скуп, јер на основу његове дефиниције мора да важи $\omega \in \Omega$. Уколико је $\Omega = \{\omega\}$, то значи да рјешење подпроблема ω није у конфликту са рјешењима осталих подпроблема, па ω, може бити премјештено из скупа U у скуп S. Наведене операције се налазе у линијама (**10**) и (**11**) псеудокода.

Ако је скуп Ω, има више од једног члана, онда је рјешење подпроблема ω у конфликту са сим подпроблемима у Ω. Тада, уклонимо све чланове скупа Ω из U и S. Затим направимо унију свих индекса промјенљивих одлучивања из скупа Ω, другим ријечима објединимо све подпроблеме из Ω у један $\cup\Omega$, и придружимо га скупу нерјешених подпроблема U. Опис овог параграфа односи се на (**10**), (**13**) и (**14**) линију псеудокода.

Када скуп U, постане празан скуп, онда су сви подпроблеми ријешени и међусобно нијесу у конфликту. Тада x садржи оптимално рјешење проблема оптимизације и заједно са вриједношћу функције критеријума $f(x)$ враћа га као резултат рада алгоритма, линија (**17**).

Из описа D&C није унапријед јасно да се алгоритам зауставља, као и то да ако се D&C зауставио да ли смо добили допустиво рјешење и ако јесмо да ли је оно оптимално. Све наведено мора се доказати, како би за резултат имали потпуну коректност D&C алгоритма.

4.3.2. Основна својства D&C

Из описа D&C није унапријед јасно да се алгоритам зауставља, као и то да ако се D&C зауставио да ли смо добили допустиво рјешење и ако јесмо да ли је оно оптимално. Све наведено мора се доказати, како би за резултат имали потпуну коректност D&C алгоритма. Претпоставка, за све леме и теорему у овом потпоглављу, је да је алгоритам, којег користимо за рјешавање ПО, означен у псеудокоду D&C са $\text{Solver}(l, Dom, f, \Xi, F, \omega)$, тотално коректан. Као што је већ раније показано то је случај са SEDA-а и SEDA+ERH.

Лема 4.3.1 *– Лема о заустављању D&C алгоритма*

Алгоритам D&C се зауставља.

Доказ.

Проблем заустављања D&C се своди на проблем да ли ***until*** петља, од линије (**4**) до линије (**16**), његовог псеудокода, достиже услов изласка из ње, тј. да

ли ће у једном кораку рада алгоритма испунити се услов $U = \emptyset$. То питање није тривијално, јер се у току извршавања петље скупу U, како одузимају, тако и додају елементи у линијама (**11**) и (**14**).

У случају када је $\Omega = \{\omega\}$, након извршавања линије (**9**) псеудокода, тада се редукује кардиналност скупа , у линији (**11**), па не постоји опасност да се уђе у бесконачну ***until*** петљу.

У случају када је $|\Omega| > 1$, након извршавања линије (**9**) псеудокода, тада поред ω, скуп Ω, садржи бар један елемент φ, такав да $\varphi \neq \omega$. Елементи скупа Ω, такође припадају или скупу U, или скупу S, на основу начина на који је скуп Ω израчунат, у линији (**9**) псеудокода.

Претпоставимо да $\varphi \in U$. Тада се у линији кода (**14**) кардиналност скупа , прво умањује бар за 2, јер $\omega, \varphi \in U$, а потом повећава за један. Из свега закључујемо да се кардиналност скупа U, смањила бар за 1, па не постоји опасност да се уђе у бесконачну ***until*** петљу.

Претпоставимо сада да $\varphi \in S$. Тада се у линији (**13**) смањује кардиналност скупа S за 1, док кардиналност скупа U, због операција у линији (**14**), у најгорем случају остаје иста. Чини се да у овом случају може доћи до бесконачног извршавања ***until*** петље, јер се кардиналност скупа U не смањује. Међутим није тако, јер се овај случај може понављати само док је скуп S непразан. Како је скуп S увијек коначан, то се овај случај може поновити највише онолико пута колика је кардиналност скупа S, тј. коначан број пута. Након тога доћи ће до смањења кардиналности скупа U, чиме се спрјечава улазак у бесконачну ***until*** петљу. **QED**.

***Лема 4.3.2** – Лема о допустивом рјешењу D&C алгоритма*

Валуација x, која је резултат рада D&C алгоритма је допустиво рјешење ПО.

Доказ.

Након заустављања D&C алгоритма скуп S садржи скупове, у валуацији x неконфликтних подпроблема, почетног ПО, који су тачно ријешени. Стога, на основу дефиниције 4.3.2 и на основу неконфликтности подпроблема у S важи:

$$(\forall \omega \in S)\ \Phi\left(x_{|\omega}\right) \tag{4.3.6}$$

$$(\forall \omega \in S)(\forall i, j \in \omega)\ \ i \neq j\ \Rightarrow\ x_i\ \rho\ x_j \tag{4.3.7}$$

$$(\forall \omega_1, \omega_2 \in S)\ \ \omega_1\ \rho_x\ \omega_2 \tag{4.3.8}$$

На основу дефиниције релације неконфликтности за подскупове скупа индекса, дефиниција 4.3.3, и горње формуле (4.3.8) непосредно изводимо:

$$(\forall \omega_1, \omega_2 \in S)(\forall i \in \omega_1)(\forall j \in \omega_2)\ \ x_i\ \rho\ x_j. \tag{4.3.9}$$

Како је $L = \cup S$, из формула (4.3.7) и (4.3.9) слиједи:

$$(\forall i, j \in L)\ \ i \neq j\ \Rightarrow\ x_i\ \rho\ x_j \tag{4.3.10}$$

Из формуле (4.3.10) и дефиниције 4.3.2 непосредно изводимо да $\Phi(x)$, тј. валуација x је допустиво рјешење. **QED**.

***Лема 4.3.3** – Лема о оптималном рјешењу D&C алгоритма*

Валуација x, која је резултат рада D&C алгоритма је оптимално рјешење ПО.

Доказ.

На основу претходне леме знамо да је валуација x допустиво рјешење. Означимо са $m_x = f(x)$ вриједност функције критеријума за допустиво рјешење x. Означимо даље и чланове скупа са $S = \{\omega_1, \omega_2, \dots, \omega_k\}$. Из описа D&C алгоритма, јасно је да је скуп S партиција скупа L. Коначно уведимо и ознаке $m_x(i)$, за вриједност функције критеријума за ω_i, $i \in \{1, \dots, k\}$.

Нека је t допустиво рјешење ПО и нека је $m_t = f(t)$, вриједност критеријумске функције за t. Такође, уведимо и ознаке $m_t(i)$, за вриједност функције критеријума за ω_i, $i \in \{1, \dots, k\}$. Из описа алгоритма D&C јасно је да важи:

$$m_x = \sum_{i=1}^{k} m_x(i) \quad \text{и} \quad m_t = \sum_{i=1}^{k} m_t(i) \tag{4.3.11}$$

Пошто су $m_x(i)$ вриједности критеријумске функције за подпроблеме за ω_i, $i \in \{1, \dots, k\}$, сљедеће неједначине су тачне:

$$(\forall i \in \{1, \dots, k\})\ m_x(i)\ \leq\ m_t(i). \tag{4.3.12}$$

Из формуле (4.3.11) непосредно закључујемо и да важи:

$$m_x = \sum_{i=1}^{k} m_x(i) \leq \sum_{i=1}^{k} m_t(i) = m_t. \tag{4.3.13}$$

Дакле, за било које допустиво рјешење t, вриједност његове функције критеријума је m_t је веће или једнако од m_x вриједности функције критеријума за допустиво рјешење x, тј. $m_x \leq m_t$. Одатле непосредно закључујемо да је x једно од оптималних рјешења. **QED**.

Примјетимо да оптимално рјешење не мора да буде јединствено. D&C, баш као SEDA и SEDA+ERH обезбјеђује проналажење једе вриједности оптималног рјешења.

Теорема 4.3.1 *– Теорема о потпуној коректности D&C алгоритма*
Алгоритам D&C је тотално коректан, ако је алгоритам, којег користимо за рјешавање проблема оптимизације, означен у псеудокоду D&C алгоритма са $\text{Solver}(l, Dom, \Xi, f, \omega)$, такође тотално коректан.

Доказ.
Претпоставимо да је функција/алгоритам $\text{Solver}(l, Dom, \Xi, f, \omega)$ тотално коректан. На основу леме 4.3.1 закључујемо да се алгоритам D&C зауставља. Даље, на основу леме 4.3.2 закључујемо је рјешење до којег дође алгоритам D&C допустиво рјешење. Коначно на основу леме 4.3.3 закључујемо је рјешење до којег дође алгоритам D&C оптимално рјешење проблема оптимизације. **QED**.

4.4. ОПИС ПРОГРАМСКЕ ИМПЛЕМЕНТАЦИЈЕ И РАЧУНАРСКИХ РЕСУРСА

Псеудокодови SEDA, SEDA+ERH и SEDA+ERH+D&C на апстрактном нивоу описују рад ових алгоритама. У њима постоје детаљи који су вриједни коментара, како са аспекта изградње и усавршавања ових алгоритама, тако и са аспекта њихове програмске имплементације. У овом поглављу биће разматрани неки од ових аспеката.

4.4.1. Питање израчунавања домена промјенљивих одлучивања

У опису алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C подразумијева се да је скуп домена промјенљивих одлучивања $Dom = \{D_1, \dots, D_l\}$ унапријед

израчунат прије почетка рада алгоритама. У конкретним имплементацијама то није нужно. Довољно је да током разматрања вриједност за неку промјенљиву одлучивања одредимо скуп домена за сљедећу, која је на реду. У случајевима када ови скупови имају велику кардиналност или су међусобно зависни, тада је ефикасније генерисати један, по један скуп домена.

Међутим, утисак је да уколико је могуће их генерисати на почетку рада алгоритма тим боље. Ако су кардиналности тих скупова релативно мале и нијесу проблем са становишта употребе меморијског простора, а то дозвољава природа проблема, онда их је боље генерисати све на почетку. То је један од разлога зашто *метод седиментације* даје добре резултате за проблем додјеле везова, који се разматра у глави 6. Са друге стране, за проблеме исказне задовољивости, који се разматрани у глави 5, ефикаснији резултати се добијају ако се домени израчунавају један по један.

4.4.2. Одржавање скупа домена

Алгоритми SEDA, SEDA+ERH и SEDA+ERH+D&C томом свог рада често посежу за подацима из скупа домена и врше операције над листама које садрже елементе, односно њихове индексе, из скупа домена. Претраживања и сортирања су веома честа, па је ефикасна имплеметација, ових помоћних задатака веома битна са становишта опште ефикасности наведених алгоритама. О овоме је било ријечи у подпоглављу 4.1.7, и рјешења која су тамо наведена базирају се на ξ листама. Међутим, неки проблеми, као што су проблеми буловске задовољивости имају веома кратке ξ листе, што за посљедицу има мању ефикасност, јер се сувише времена троши на непотребно одржавање структура листи. У том случају треба изабрати структуре ограничених низова, којима је лакше управљати.

4.4.3. Питање редослиједа разматрања промјенљивих одлучивања

Ово питање је кључно за ефикасност алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C. Већ је показано колико је важно за почетни редослијед, по којем алгоритми разматрају вриједности промјенљивих одлучивања приликом излагања хеуристике *„проџени и преуреди“*. Ово питање се може поставити на два начина, као већ поменуто питање почетног редослиједа, као и то да ли мијењати распоред за неразматране промјенљиве одлучивања? У примјерима који слиједе, након почетног утврђивања, распоред није мијењан, што је можда разлог појаве *„тешких примјера“*, који се дуго

рјешавају. Модификација наведених алгоритама, која би омогућила промјену распореда неразматраних промјенљивих одлучивања је релативно једноставна, што чини ово питање интересантним за будућа истраживања.

4.4.4. Рекурзивна или нерекурзивна имплементација

Алгоритми SEDA, SEDA+ERH и SEDA+ERH+D&C су изложени у рекурзивној форми. Међутим, њихова имплементација, која је коришћена за добијање екперименталних резултат је била нерекурзива. Она је тежа за излагање, али ефикаснија када су у питању рачунарски ресурси: процесорско вријеме и меморија. Такође, изискује додатан програмерски напор за контролу и одржавање методе *„враћања по трагу“ (Backtracking)*, која је *„скривена“* у рекурзивној формулацији алгоритама. Неефикасна имплеметација овог механизма може успорити рад рјешавача, умјесто да за резултат има убрзање. Само питање управљања стеком *(stack)* је веома важни са аспекта ефикасности и у случају нерекурзивне имплеметације треба му посветити посебну пажњу.

4.4.5. Програмерски детаљи

Метода седиментације је имплементирана у два програмска језика. Прво у *Wolfram Mathematica*-и, а затим у прогрмском језику *C*. Укупна дужина кода у *Wolfram Mathematici* је 1000 линија. Имплементација у програмском језику *C* подјељена је у три цјелине: учитавање улазних података и генерисање извјештаја, има 155 линија кода; програмски модуо који садржи дефиниције, функције и процедуре везане за рад са доменима промјенљивих одлучивања, има 550 линија кода и управљачки механизам рјешавача са имплементираним SEDA, SEDA+ERH и SEDA+ERH+D&C алгоритмима, који има 1519 линија кода. Дакле, *C* имплементација има укупно 2224 линија кода. Имплементација методе у *C*, се може оцијенити као средње тешка, будући да поред самог механизма наведених алгоритама, доста кода који одржава структуре података потребне за рад алгоритама. Због тога је код у *Wolfram Mathematica*-и скоро дупло краћи, јер у овом програмском језику постоје бројне уграђене рутине, за рад са низовима, које поједностављују писање програма. Приближни однос брзина рјешавања, истог проблема, је 250 пута у корист *C* имплементације у односу на *Wolfram Mathematica*-а имплементацију.

Прилагођење кода, било да се ради о у *Wolfram Mathematica*-и или *C*-у, за другу врсту проблем није једноставно и захтјева поприлично вријеме. То се нарочито односи на *C* имплементацију.

5. ПРИМЈЕНА МЕТОДА СЕДИМЕНТАЦИЈЕ НА ПРОБЛЕМЕ МАТЕМАТИЧКЕ ЛОГИКЕ И АЛГЕБРЕ

Проблеми математичке логике на којим ћемо приказати примјену *методе седиментације* су проблеми *Буловске задовољивости* формула исказног рачуна. Ови проблеми су у литератури познати као SAT и Max-SAT проблеми.

5.1. SAT И MAX-SAT ПРОБЛЕМИ

Основни појмови *исказног рачуна* подразумијевамо да су познати, па нећемо наводити њихову дефиницију, која се може наћи у бројној литератури: (Петровић & Мијајловић, 2012). Дефинисаћемо само појмове, потребне за лакше праћење SAT и Max-SAT проблема.

Дефиниција 5.1.1 *– Исказне промјењиве и литерал*

Нека је дат скуп *буловских промјењивих* $V = \{p_1, \dots, p_m\}$. За исказну промјењиву $p_i \in V$ дефинишемо њен *литерал* као p_i или $\neg p_i$.

Дефиниција 5.1.2 *– Клауза, празна клауза*

Клауза C је формула у облику дисјункције литерала: $C = \lambda_1 \vee \lambda_2 \vee \dots \lambda_k$. Празну клаузу означавамо са $\emptyset$.

Дефиниција 5.1.3 *– CNF формула*

Формула φ је у *конјунктивној нормалној форми,* на енглеском језику *Conjunctive Normal Form* (CNF), ако је облика $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_l$, гдје су $C_1, C_2 \dots C_l$ клаузе. У контексту SAT-а и Max-SAT-а, понекад се формуле у конјунктивној нормалној форми називају и *SAT формулама*. Скуп свих исказних промјењивих, које се појављују у формули φ, означаваћемо са $\mathrm{var}(\varphi)$.

Доказ да се свака формула исказног рачуна има себи еквивалентну формулу у CNF-у је изостављен.

Дефиниција 5.1.4 *– Клауза са тежином*

Уређен пар (C, w), гдје је C клауза и w је природан број или бесконачно, називамо *клаузом са тежином*. Природан број w изражава *„казну“* уколико

клауза C није тачна. Клаузу са тежином зовемо *тешком*, уколико јој њој одговарајућа казна је бесконачно, иначе је зовемо *лаком*.

Дефиниција 5.1.5 *– CNF формула са тежинама*

Скуп клауза са тежинама:.

$$\varphi = \{(C_1, w_1), \dots, (C_k, w_k), (C_{k+1}, \infty), \dots, (C_{k+k'}, \infty)\}, \tag{5.1.1}$$

гдје су првих k клауза лаке, а посљедњих k' тешке зовемо CNF формулом са тежинама. CNF формули φ са тежинама одговара CNF формула $\bar{\varphi} = \bigwedge_{i=1}^{k+k'} C_i$.

За CNF формулу са тежинама φ, дефинишемо њен *лаки дио* као $\varphi_{\text{soft}} = \{(C_1, w_1), \dots, (C_k, w_k)\}$ и њен *тешки дио* као $\varphi_{\text{hard}} = \{(C_{k+1}, \infty), \dots, (C_{k+k'}, \infty)\}$.

Очигледно, важи $\varphi = \varphi_{\text{soft}} \cup \varphi_{\text{hard}}$, као и $\bar{\varphi} = \bar{\varphi}_{\text{soft}} \wedge \bar{\varphi}_{\text{hard}}$.

Из наведених дефиниција 5.1.3 и 5.1.5 јасно је да CNF формулу $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_l$ можемо посматрати као CNF формулу са тежинама:

$$\dot{\varphi} = \{(C_1, 1), \dots, (C_l, 1)\}. \tag{5.1.2}$$

Дефиниција 5.1.6 *– Валуација исказних промјењивих*

Функција облика $v: \{1, \dots, m\} \longrightarrow \{0,1\}$, којом свакој исказној промјењивој p_i, $i \in \{1, \dots, m\}$ додјељује вриједност $v_i(p_i)$, зовемо *валуацијом исказних промјењивих*. Сматрамо да је вриједност 1, представља тачно, а вриједност 0 нетачно.

Дефиниција 5.1.7 *– Тачност литерала у валуацији*

Нека је $v: \{p_1, \dots, p_m\} \longrightarrow \{0,1\}$ валуација исказних промјењивих. Дефиницију валуације v проширујемо на литерале λ на сљедећи начин:

$$v(\lambda) = \begin{cases} v(p_i) & : \quad \lambda = p_i, \ \text{ за неко } i \in \{1, \dots, m\}, \\ 1 - v(p_i) & : \quad \lambda = \neg p_i, \ \text{ за неко } i \in \{1, \dots, m\}. \end{cases} \tag{5.1.3}$$

Дефиниција 5.1.8 *– Тачност клаузе у валуацији*

Нека је $v: \{p_1, \dots, p_m\} \longrightarrow \{0,1\}$ валуација исказних промјењивих. Дефиницију валуације v проширујемо на клаузе $C = \lambda_1 \vee \lambda_2 \vee \dots \lambda_k$ на сљедећи начин:

$$v(C) = \begin{cases} 1 & : \sum_{i=1}^{k} v(\lambda_i) > 0, \\ 0 & : \sum_{i=1}^{k} v(\lambda_i) = 0. \end{cases} \tag{5.1.4}$$

Дефиниција 5.1.9 *– Тачност CNF формуле у валуацији*

Нека је $v: \{p_1, \dots, p_m\} \longrightarrow \{0,1\}$ валуација исказних промјењивих. Дефиницију валуације v проширујемо на CNF формуле $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_l$ на сљедећи начин:

$$v(\varphi) = \prod_{i=1}^{l} v(C_i). \tag{5.1.5}$$

Након уведених основних дефиниција могуће је дефинисати проблеме SAT, Max-SAT и његове варијанте.

Дефиниција 5.1.10 *– cost функција*

За CNF сваку формулу са тежинама $\varphi = \{(C_1, w_1), \dots, (C_l, w_l)\}$, и сваку валуацију њених исказних промјењивих $v: \{p_1, \dots, p_m\} \longrightarrow \{0,1\}$, дефинишемо фукцију cost, као:

$$\text{cost}(\varphi, v) = \sum_{i=1}^{l} w_i\big(1 - v(C_i)\big). \tag{5.1.6}$$

Из дефиниције 5.1.10, јасно је да функцију $\text{cost}(\varphi, v)$ рачунамо као збир тежина свих нетачних клауза формуле φ, у валуацији v. Да би исправно могли израчунати суму у формули (5.1.6), подразумијевамо да је $0 \cdot \infty = 0$.

Дефиниција 5.1.11 *– Проблем исказне задовољивости (SAT проблем)*

Проблем исказне задовољивости, на енглеском језику *Boolean Satisfiability Problem* (SAT), дефинишемо као проблем одређивања да ли је формула исказног рачуна која је у CNF облику задовољива, тј. да ли је тачна у некој валуацији. У овом контексту ограничавамо се на CNF формуле, па дефиницију SAT проблема за CNF формулу φ можемо формулисати као:

$$(\exists v{:}\text{var}(\varphi) \to \{0,1\})\ \ v(\dot{\varphi}) = 1. \tag{5.1.7}$$

Њој еквивалентна формулација, користећи cost функцију је:

$$(\exists v{:}\mathrm{var}(\varphi) \to \{0,1\})\ \ \mathrm{cost}(\dot{\varphi}, v) = 0. \tag{5.1.8}$$

Дефиниција 5.1.12 *– Проблем Max-SAT*

Проблем максималне задовољивости формуле, на енглеском језику *Maximum Satisfiability Problem* (Max-SAT), дефинишемо као проблем одређивања валуације v, тако да се максимизује број тачних клауза у CNF формули φ или еквивалентно да се минимизује број нетачних клауза у CNF формули φ. Имајући у виду дефиниције у овом потпоглављу, Max-SAT проблем можемо дефинисати као:

$$\min_{v:\mathrm{var}(\varphi)\to\{0,1\}} \mathrm{cost}(\dot{\varphi}, v). \tag{5.1.9}$$

Дефиниција 5.1.13 *– Тежински Max-SAT проблем*

Проблем максималне тежинске задовољивости формуле, на енглеском језику *Weighted Maximum Satisfiability Problem* (W Max-SAT), дефинишемо као проблем одређивања валуације v, тако да се максимизује сума тежина тачних клауза у лакој CNF формули са тежинама φ или еквивалентно да се минимизује сума тежина нетачних клауза у лакој CNF формули са тежинама φ. Имајући у виду дефиниције у овом потпоглављу, овај проблем можемо дефинисати као:

$$\min_{v:\mathrm{var}(\varphi)\to\{0,1\}} \mathrm{cost}(\varphi, v). \tag{5.1.10}$$

У *тежинском* Max-SAT проблему подразумијева се да CNF формула φ, садржи само лаке клаузе, а не садржи тешке, односно за њу важи $\varphi = \varphi_{\mathrm{soft}}$ и $\varphi_{\mathrm{hard}} = \emptyset$.

Дефиниција 5.1.14 *– Парцијални Max-SAT проблем*

Проблем максималне парцијалне задовољивости формуле, на енглеском језику *Partial Maximum Satisfiability Problem* (P Max-SAT), дефинишемо за CNF формуле са тежинама φ, такве да $\varphi_{\mathrm{soft}} = \{(C_1, 1), \dots, (C_k, 1)\}$ и $\varphi_{\mathrm{hard}} \neq \emptyset$ као проблем одређивања валуације v, тако да се минимизује број нетачних клауза у CNF формули са тежинама φ. Имајући у виду дефиниције у овом потпоглављу, овај проблем можемо дефинисати као:

$$\min_{v:\mathrm{var}(\varphi)\to\{0,1\}} \mathrm{cost}(\varphi, v). \tag{5.1.11}$$

Примјетимо да у P Max-SAT проблему рјешење, уколико постоји мора да задовољава сваку тешку клаузу формуле φ, што не мора да важи и за лаке клаузе формуле φ.

Дефиниција 5.1.15 *– Парцијални тежински Max-SAT проблем*

Проблем максималне парцијалне тежинске задовољивости формуле, на енглеском језику *Partial Weighted Maximum Satisfiability Problem* (PW Max-SAT), дефинишемо за CNF формуле са тежинама φ, такве да $\varphi_{\text{soft}} \neq \emptyset$ и $\varphi_{\text{hard}} \neq \emptyset$ као проблем одређивања валуације v, тако да се минимизује сума тежина нетачних клауза у CNF формули са тежинама φ. Имајући у виду дефиниције у овом потпоглављу, овај проблем можемо дефинисати као:

$$\min_{v:\text{var}(\varphi)\to\{0,1\}} \text{cost}(\varphi, v). \tag{5.1.12}$$

Слично као и код P Max-SAT проблема, рјешење PW Max-SAT проблема, уколико постоји мора да задовољава сваку тешку клаузу формуле φ, што не мора да важи и за лаке клаузе формуле φ.

Примјетимо на крају да се у овој формулацији Max-SAT проблем, тежински Max-SAT проблем, парцијални Max-SAT проблем и парцијално тежински Max-SAT проблем, заправо исти проблем са гледишта услова минимизације. Формуле (5.1.9—12) су исте формуле, оно што је различито јесте за какав облик CNF формуле се рјешава проблем.

У наставку, све проблеме дефинисане у овом потпоглављу зовемо *SAT и Max-SAT проблемима*. Када то буде потребно, дискутујемо сваки од њих појединачно.

Проблеми SAT и Max-SAT у којима се разматрају формуле у чијим се клаузама јавља тачно k литерала издвајамо у посебну класу SAT-у и Max-SAT-у. Ту врсту проблема називамо kSAT и Max-kSAT проблемима.

SAT проблем је по комплексности *NP-complete* проблем (Cook, 1970). Исто важи и за Max-SAT проблем, јер се он може свести на SAT проблем.

5.2. ПРЕГЛЕД ПОСТОЈЕЋИХ МЕТОДА ЗА РЈЕШАВАЊЕ MAX-SAT ПРОБЛЕМА

Будући да SAT и Max-SAT примјер у овог глави служи превасходно као илустрација примјене *методе седиментације,* слиједи кратак преглед радова из овог домена.

Већина комплетних приступа рјешавању SAT проблема базира се на *„тражи и ограничи“* приступу и *„враћања по трагу“* механизму. Дакле истим оним, које користи и *метод седиментације*. Алгоритам за рјешавање SAT проблема добио је име по својим творцима: *Давис-Путнам-Логеман-Ловеланд* (*Davis–Putnam–Logemann–Loveland,* DPLL), (Davis & Putnam, 1960), (Davis, Logemann, & Loveland, 1962). Уз основни алгоритам развијене су бројне технике које су усавршиле основни алгоритам, као што су: *„скок уназад“, „поновно започињање“, „учење вођеном конфликтом“,* итд. Такође, боље технике имплементације рјешавача, довеле су такође до унапрјеђења ефикасности DPLL алгоритма. Опширни прикази SAT алгоритама дати су у (Biere, Heule, Van Maaren, & Walsh, 2009) и (Marić, 2009).

Два су главна приступа рјешавању Max-SAT проблема: *„тражи и ограничи“* и приступ базиран на *(не)задовољивости*. Оцјена је да су методи базирани на *„тражи и ограничи“* принципу за случајно генерисане проблеме (CNF формуле) ефикаснији (Li, Manya, Mohamedou, & Planes, 2009), (Heras, Larrosa, & Oliveras, 2007) и (Lin, Su, & Li, 2008). За разлику од њих, методи базирани на (не)задовољивости ефикаснији код индустријских и реалних проблема (Ansotegui, Bonet, & Levy, 2013) и (Fu Z. , 2009).

5.3. КОМБИНАТОРНА ФОРМУЛАЦИЈА SAT И MAX-SAT ПРОБЛЕМА ЗА МЕТОД СЕДИМЕНТАЦИЈЕ

SAT и Max-SAT проблеме могуће је моделовати на два начина да би примијенили *метод седиментације* за њихово рјешавање. Први начин полази од клауза CNF формуле, а други од промјенљивих CNF формуле. Наведена су оба моделовања, а затим упоредити њихову ефикасност у поглављу са експерименталним резултатима.

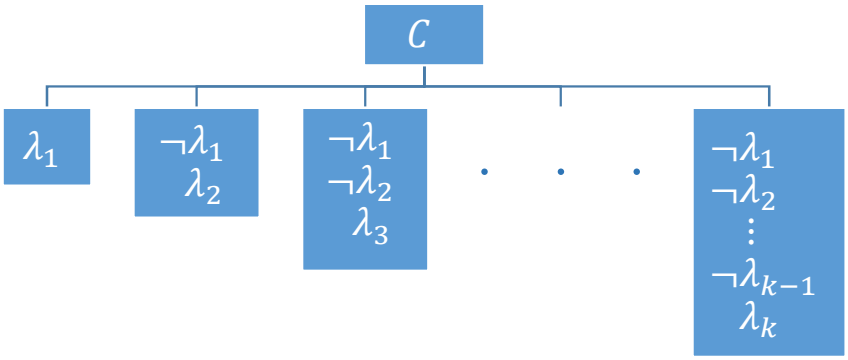
5.3.1. SAT/Max-SAT моделовање путем клауза

Нека је дата CNF формула $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_l$. Означимо са $C = \lambda_1 \vee \lambda_2 \vee \ldots \lambda_k$, било коју клаузу CNF формула φ. Примјетимо да важе еквиваленције:

$$C \Leftrightarrow \lambda_1 \vee (\neg\lambda_1 \wedge \lambda_2) \vee (\neg\lambda_1 \wedge \neg\lambda_2 \wedge \lambda_3) \vee \ldots \vee (\neg\lambda_1 \wedge \neg\lambda_2 \wedge \ldots \wedge \neg\lambda_{k-1} \wedge \lambda_k), \tag{5.3.1}$$

$$\neg C \Leftrightarrow \neg\lambda_1 \wedge \neg\lambda_2 \wedge \ldots \wedge \neg\lambda_{k-1} \wedge \neg\lambda_k. \tag{5.3.2}$$

На основу њих испитивање да ли је тачна клаузе C, можемо извести са сљедећи начин: клаза C, је тачна ако је λ_1 тачно, или ако је $\neg\lambda_1 \wedge \lambda_2$, тачно, и наставимо тако до $\neg\lambda_1 \wedge \neg\lambda_2 \wedge \ldots \wedge \neg\lambda_{k-1} \wedge \lambda_k$. Уколико клауза C, није тачна, то може бити једино ако је $\neg\lambda_1 \wedge \neg\lambda_2 \wedge \ldots \wedge \neg\lambda_{k-1} \wedge \neg\lambda_k$ тачно. Ако идући редом по клаузама кренемо да конструишемо стабло, гранајући га по горе наведеним гранам, водећи рачуна о конзистентност сваке гране, добијамо поступак, којим можемо да утврдимо тачност CNF формула φ.



Илустрација 5.3.1 *– Приказ гранања, у случају када је клауза* $C = \lambda_1 \vee \lambda_2 \vee \ldots \lambda_k$ *тачна.*

Дефиниција 5.3.1 *– Функција* $\mathrm{oc}(\varphi, \lambda)$

Означимо са $\mathrm{oc}(\varphi, \lambda)$ број појављивања исказне промјењиве из литерала λ, у формули φ.

Дефиниција 5.3.2 *– Домен тачности клаузе C*

За клаузу $C = \lambda_1 \vee \lambda_2 \vee \ldots \lambda_k$, такву да $\mathrm{oc}(\varphi, \lambda_1) \geq \mathrm{oc}(\varphi, \lambda_2) \geq \cdots \geq \mathrm{oc}(\varphi, \lambda_k)$, дефинишемо њен *домен тачности* Δ_C као низ низова:

$$\Delta_C = \langle\langle\lambda_1\rangle, \langle\neg\lambda_1, \lambda_2\rangle, \langle\neg\lambda_1, \neg\lambda_2, \lambda_3\rangle, \ldots, \langle\neg\lambda_1, \neg\lambda_2, \ldots, \neg\lambda_{k-1}, \lambda_k\rangle\rangle. \tag{5.3.3}$$

Означимо са $\mathrm{oc}(\varphi, \lambda)$ број појављивања исказне промјењиве из литерала λ, у формули φ.

Дефиниција 5.3.3 *– Потпуни домен тачности клаузе C*

За клаузу $C = \lambda_1 \vee \lambda_2 \vee \ldots \lambda_k$, такву да $\mathrm{oc}(\varphi, \lambda_1) \geq \mathrm{oc}(\varphi, \lambda_2) \geq \cdots \geq \mathrm{oc}(\varphi, \lambda_k)$, дефинишемо њен *потпуни домен тачности* $\overline{\Delta}_C$ као низ низова:

$$\overline{\Delta}_C = \Delta_C + \langle \neg\lambda_1, \neg\lambda_2, \dots, \neg\lambda_{k-1}, \neg\lambda_k \rangle. \tag{5.3.4}$$

Операција сабирања у формули (5.3.4) је операција конкатенирања низова. Такође подразумијева се да у листама које су елементи Δ_C и $\overline{\Delta}_C$ примјењује правило двоструке негације над литералима који их чине.

Након увођења дефиниција 5.3.2 и 5.3.3 можемо дефинисати улазне податке, функције и процедуре за SEDA, како је то описано у глави 4.

Подразумијева се да је потребно ријешити проблем SAT или неки од Max-SAT проблема за CNF формулу са тежинама φ. Уколико се ради баш о SAT или Max-SAT проблемима, онда њихову рјешавамо CNF формулу са тежинама $\dot{\varphi}$, дефинисану у формули (5.1.2).

Скуп *промјенљивих одлучивања* $X = \{x_1, \dots, x_l\}$.

Скуп домена $Dom = \{D_1, \dots, D_l\}$ у случају SAT-а је $D_i = \Delta_{C_i}$, $i \in L$.

Скуп домена $Dom = \{D_1, \dots, D_l\}$ у случају Max-SAT-а и W Max-SAT-а је $D_i = \overline{\Delta}_{C_i}$, $i \in L$. За разлику од SAT-а, у случају Max-SAT-а и W Max-SAT-а клаузе могу бити нетачне, па су литерали из потпуног домена тачности $\overline{\Delta}_{C_i}$ клаузе C_i.

У случају P Max-SAT-а и PW Max-SAT-а, *скуп домена* за клаузу C_i дефинишемо као $D_i = \Delta_{C_i}$, ако је $w_i = \infty$, иначе, ако је $w_i < \infty$, тада га дефинишемо као $D_i = \overline{\Delta}_{C_i}$, за $i \in L$.

Функција критеријума f је идентична функцији cost, која је дефинисана у дефиницији 5.1.10.

Скуп хеуристичких релација $\Xi = \{\prec_1, \dots, \prec_l\}$, који садржи хеуристике тоталног уређења са скупове домена, поклапа се са поретком елемената у скуповима домена. Скупови домена су овдје уређене листе, сходно дефиницијама 5.3.2 и 5.3.3.

Ради лакшег формулисања функције завршетка конструкције означимо са $V[x_1, x_2, \dots, x_\theta]$ парцијалну валуацију скупа исказних промјенљивих у θ кораку конструкције. Вриједност функције завршетка конструкције $F(\theta)$ је једнака *тачно*, уколико је домен функције $V[x_1, x_2, \dots, x_\theta]$ једнак скупу исказних промјенљивих, иначе је *нетачно*.

Процедуре и функције: FindSolution(θ, Dom), Estimation(θ, Dom), NonDetVarEstimation(θ), ReportSolution() и CutOff(θ, ε) су управо онакве какве су описане у потпоглављу 4.1.3. Функција NonDetVarEstimation(θ) враћа суму $w_i\big(1 - V[x_1, x_2, \dots, x_\theta](C_i)\big)$, за оне клаузе C_i, $i \in \{\theta + 1, \dots, l\}$, за које је могуће израчунати $V[x_1, x_2, \dots, x_\theta](C_i)$.

Функција Sediment(θ, x) брише све низове литерала у доменима неодређених промјенљивих одлучивања које нијесу сагласне са $V[x_1, x_2, \dots, x_\theta]$. Њој се могу додати бројне хеуристике које постоје за рјешавање SAT и Max-SAT проблема. У приказу експерименталних резултата приказаће се утицај правила чистог литерала, на енглеском *Pure Literal Rule* (PLR) на ефикасност SEDA. PLR можемо кратко описати као технику у којој се испитује да ли се у формули φ, или у њеним поједностављењима, јавља изолован литерал λ, уколико постоји такав литерал, онда треба додјели исказној промјенљивој, која се у њему јавља, такву вриједност, да тај литерал буде тачан.

Из описаног моделовања SAT и Max-SAT проблема путем клауза за *метод седиментације* непосредно изводимо да он задовољава све услове из поглавља 4.1. То за посљедицу има да комбиновањем овог моделовања SAT и Max-SAT проблема са алгоритмима SEDA и SEDA+ERH добијамо два рјешавача наведених проблема.

5.3.2. SAT/Max-SAT моделовање путем промјенљивих

Уведимо SAT/Max-SAT моделовање путем промјенљивих, тако што за сваку исказну промјенљиву из скупа $V = \{p_1, \dots, p_m\}$ придружимо њој одговарајућу *промјенљиву одлучивања* из скупа $X = \{x_1, \dots, x_l\}$. У овом случају, јасно је да $m = l$, па надаље користимо l, као број исказних промјенљивих и као број промјенљивих одлучивања.

Скуп домена $Dom = \{D_1, \dots, D_l\}$, тада је $D_i = \{0,1\}$, $i \in L$, јер исказна промјенљива може бити или тачна, или нетачна.

Дефиниција 5.3.4 *– Функција* Simplify(φ, θ, d)

Функција Simplify(φ, i, d) врши сва поједностављења формуле φ, ако исказна промјенљива p_i узме вриједност d.

За формулацију скупа хеуристичких релација $\Xi = \{\prec_1, \dots, \prec_l\}$, претпостављамо да је у кораку θ, конструкције рјешења доступан сљедећи рекурентни низ формула:

$$\begin{aligned} &\varphi_1 \equiv \varphi, \\ &\varphi_{i+1} \equiv \text{Simplify}(\varphi_i, i, x_i), \quad i \in \{1, \dots, l-1\}. \end{aligned} \tag{5.3.5}$$

Тада се хеуристичке релације $\prec_i$, $i \in L$ могу дефинисати за свако $a, b \in D_i$, $i \in L$, као:

$$a \prec_i b \;\Leftrightarrow\; a = 0 \wedge b = 1 \wedge \text{oc}(\varphi_i, \neg p_i) > \text{oc}(\varphi_i, p_i). \tag{5.3.6}$$

Из формуле (5.3.6) закључујемо да ће вриједност нетачно, тј. 0, бити прије разматрана од вриједност тачно, тј. 1, за исказну промјењиву p_i, једино ако је број појављивања литерала $\neg p_i$ већи од броја појављивања литерала p_i у формули φ_i.

Функцију критеријума f, дефинишемо као број тачних клауза као у потпоглављу 4.1.1, тако што дефинишемо функције f_i, $i \in L$ као број тачних клауза у формули φ_i, ако је $p_i = x_i$.

Из овако дефинисане функције критеријума лако изводимо функције $\text{Estimation}(\theta, Dom)$ и $\text{NonDetVarEstimation}(\theta)$.

Процедуре и функције: $\text{FindSolution}(\theta, Dom)$, $\text{ReportSolution}()$ и $\text{CutOff}(\theta, \varepsilon)$ су управо онакве какве су описане у потпоглављу 4.1.3.

Функција $\text{Sediment}(\theta, x)$ врши пропагирање додјеле $p_\theta = x_\theta = x$ у формули φ_θ, како је то описано у формули (5.3.5).

Из описаног моделовања SAT и Max-SAT проблема путем промјењивих за *метод седиментације* непосредно изводимо да он задовољава све услове из поглавља 4.1. То за посљедицу има да комбиновањем овог моделовања SAT и Max-SAT проблема са алгоритмима SEDA и SEDA+ERH добијамо два рјешавача наведених проблема.

Моделовање моделовања SAT и Max-SAT проблема путем промјењивих за *метод седиментације* је еквивалентан DPLL алгоритму за рјешавање SAT и Max-SAT проблема.

5.4. ЕКСПЕРИМЕНТАЛНИ РЕЗУЛТАТИ

У овом поглављу биће приказани упоредни резултати три рјешавача базираних на методи седиментације, описаних у поглављу 5.3. Биће испитани следећи рјешавачи:

1. MAX-SAT рјешавач заснован на моделовању путем клаузан, означићемо га са CM;
2. MAX-SAT рјешавач заснован на моделовању путем клаузан, коме је додато правило чистог литерала PLR, означићемо га са CM+PLR и
3. MAX-SAT рјешавач заснован на моделовању путем промјенљивих, означићемо га са VM.

Сви рјешавачи су кодирани у *Wolfram Mathematica v10.3* програмском језику. Будући да су ови резулати добијени преко програмског језика који се интерпретира, резултати се не могу поредити са *state of art* рјешавачима SAT и Max-SAT проблема. Примјер Max-2SAT и Max-3SAT, који се наводе у наставку су дат ради приказивања примјенљивости *методе седиментације*. Такође, сврха им је да покажу колико различита моделовања, једног истог проблема, могу да утичу на ефикасност рјешавања, као што је то случај са CM и VM у случају Max-SAT проблема.

Тестови су изведени на рачунару са *Intel Core i7 Q720* 1.60-GHz процесором, 6 GB RAM меморије, којим је управљао *Microsoft Windows 8* 64-bit оперативни систем.

5.4.1. Опис тест примјера

Тестови су изведени за Max-2SAT и Max-3SAT проблеме. Примјери за Max-2SAT проблем имају 30 исказних промјенљивих, док су примјери за Max-3SAT проблем имају 50 исказних промјенљивих. Разматрани су посебно примјери формула са 25, 50, 75, 100, 125, 150, 175 и 200 клауза, за сваки од Max-2SAT і Max-3SAT проблема. Такође, за сваку од наведених група клауза испитано је по 300 примјера.

У табелама приказана су минимална времена, просјечна времена и максимална времена рјешавања, за сваки алгоритам који смо навели на почетку овог поглавља. Поред наведених времена израженог у секундама, дата је и стандардна девијација времена рјешавања за сваку групу клауза.

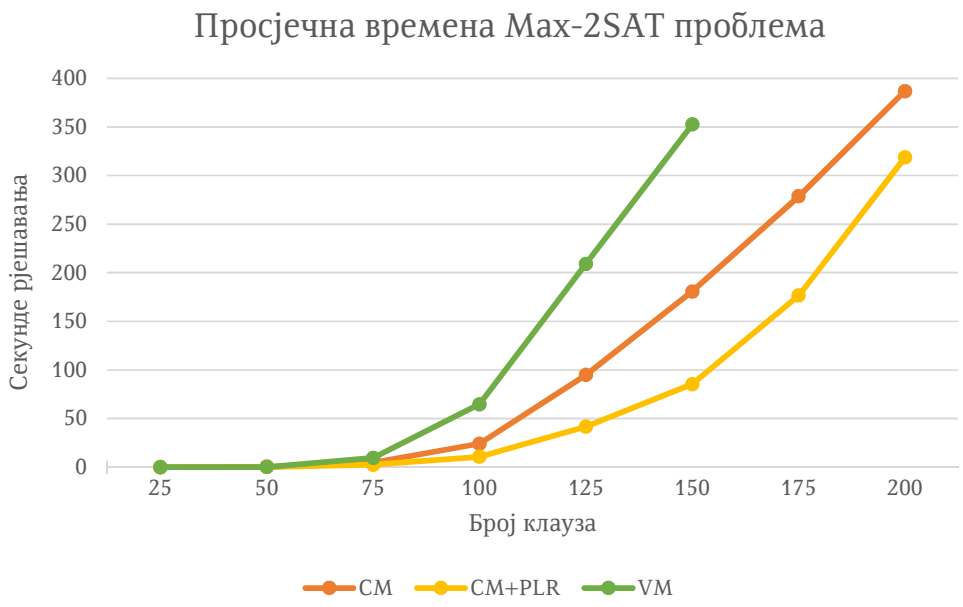
5.4.2. Међусобно поређење предложених алгоритама

У табели 5.4.1 дата су упоредна времена рјешавања CM, CM+PLR и VM алгоритама за рјешавање Max-2SAT проблема са 30 исказних промјенљивих. CM моделовање је у просјеку брже од VM моделовања. VM моделовање у примјерима формула са 150 клауза има просјечно вријеме рјешавања 352.66 секунди, CM-а 180.67, а CM+PLR-а 85.36 секунди. У том случају CM је **1.95** пута бржи од VM-а, а CM+PLR је **4.13** пута бржи од VM-а. Такође, CM+PLR је **2.12** пута бржи од CM-а.

За више од 150 клауза VM постаје превише спор, па тестови нијесу вршени. За формуле са 175 и 200 клауза CM+PLR је и даље бржи од CM-а, али се та предност смањује. У случају формула са 200 клауза CM+PLR је само **1.21** пута бржи од CM-а. Просјечна времена рјешавања за проблем Max-2SAT приказана су на илустрацији 5.4.1.

Табела 5.4.1 *– Упоредна времена извршавања тестова алгоритама CM, CM+PLR и VM за Max-2SAT са 30 исказних промјенљивих*

	Max-2SAT | 30 пром. | 300 примјера											
	CM				*CM+PLR*				*VM*			
l	min	**avg**	max	σ	min	**avg**	max	σ	min	**avg**	max	σ
25	0.03	**0.06**	0.14	0.01	0.01	**0.02**	0.08	0.01	0.02	**0.06**	0.09	0.01
50	0.12	**0.47**	4.58	0.49	0.06	**0.23**	1.09	0.16	0.06	**0.26**	2.29	0.26
75	0.31	**4.64**	37.53	4.69	0.22	**2.48**	13.35	2.08	0.30	**9.58**	114.50	12.87
100	1.84	**24.18**	176.11	20.09	0.80	**10.66**	36.22	7.28	1.47	**64.74**	583.08	70.86
125	6.81	**94.85**	478.22	64.06	1.89	**41.69**	177.12	31.88	8.80	**208.98**	1624.01	220.68
150	27.11	**180.67**	712.38	116.06	7.94	**85.36**	540.57	61.20	27.63	**352.66**	1767.90	280.64
175	28.42	**278.69**	1261.10	179.36	32.35	**176.52**	696.87	112.00	–	–	–	–
200	75.77	**386.73**	1564.56	226.96	27.64	**318.62**	1188.21	177.67	–	–	–	–



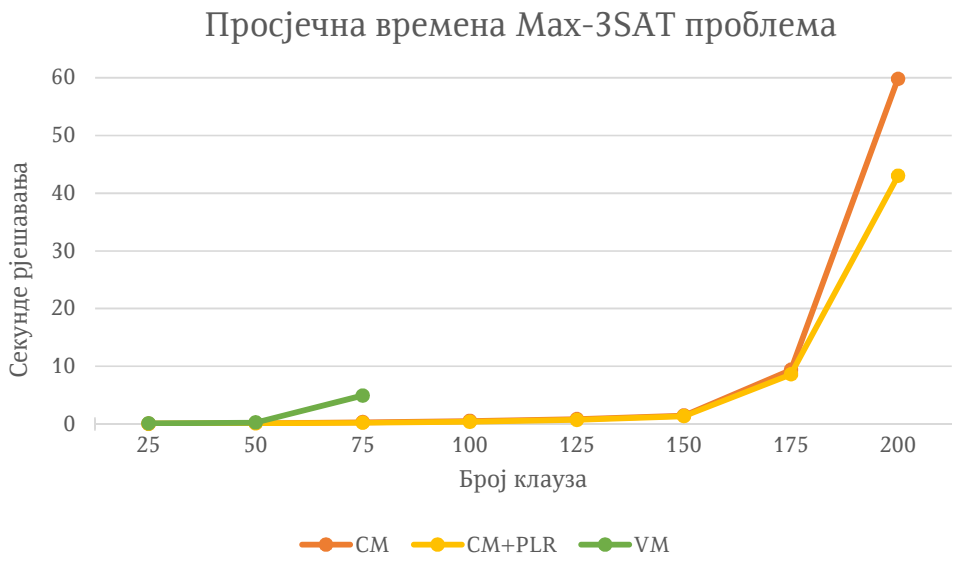
Илустрација 5.4.1 *– Просјечне брзине рјешавање Max-2SAT проблема.* На хоризонталној оси се налази број клауза, док се на вертикалној оси налазе просјечна времена рјешавања проблема.

Табела 5.4.2 *– Упоредна времена извршавања тестова алгоритама CM, CM+PLR и VM за Max-3SAT проблем са 50 исказних промјењивих*

	Max-3SAT | 50 пром. | 300 примјера											
	CM				*CM+PLR*				*VM*			
l	min	**avg**	max	σ	min	**avg**	max	σ	min	**avg**	max	σ
25	0.02	**0.05**	0.08	0.01	0.02	**0.02**	0.05	0.01	0.05	**0.09**	0.20	0.03
50	0.11	**0.16**	0.33	0.03	0.05	**0.09**	0.16	0.02	0.11	**0.25**	0.45	0.07
75	0.22	**0.30**	0.53	0.05	0.12	**0.21**	0.35	0.04	0.17	**4.93**	216.14	21.16
100	0.34	**0.52**	1.05	0.13	0.23	**0.39**	0.85	0.08	–	–	–	–
125	0.51	**0.82**	3.57	0.32	0.41	**0.69**	2.79	0.23	–	–	–	–
150	0.63	**1.46**	17.84	1.44	0.58	**1.36**	15.75	1.37	–	–	–	–
175	0.81	**9.38**	181.55	21.20	1.21	**8.59**	135.92	16.99	–	–	–	–
200	1.01	**59.79**	1691.97	123.28	1.04	**43.00**	371.05	58.30	–	–	–	–

У табели 5.4.2 дата су упоредна времена рјешавања CM, CM+PLR и VM алгоритама за рјешавање Max-3SAT проблема са 50 исказних промјењивих. CM моделовање је у просјеку брже од VM моделовања. VM моделовање у примјерима формула са 75 клауза има просјечно вријеме рјешавања 4.93 секунде, CM 0.30 секунди, а CM+PLR 0.21 секунду. У том случају CM је **1.95** пута бржи од VM-а, а CM+PLR је **16.43** пута бржи од VM-а. Такође, CM+PLR је **23.48** пута бржи од CM-а.

За више од 75 клауза VM постаје превише спор, па тестови нијесу вршени. За формуле са 100 и 200 клауза CM+PLR је и даље бржи од CM-а, али знатно мање него што је то био случај са Max-2SAT проблемом. Разлог томе је учесталија појава чистих литерала у Max-2SAT проблему него у Max-3SAT проблему. У случају формула са 200 клауза CM+PLR је **1.39** пута бржи од CM-а. Просјечна времена рјешавања за проблем Max-3SAT приказана су на илустрацији 7.4.1.



Илустрација 5.4.2 *– Просјечне брзине рјешавање Max-3SAT проблема.* На хоризонталној оси се налази број клауза, док се на вертикалној оси налазе просјечна времена рјешавања проблема.

Из тест примјери изводимо неколико закључака.

1. CM моделовање за Max-SAT проблеме је ефикасније од VM-а у контексту *методе седиментације*.
2. Додавање хеуристика, у овом случају PLR, повећава ефикасност алгоритама за рјешавање Max-SAT проблема. Будући да постоје бројне хеуристике за убрзање рјешавања проналажење и имплементацијом одговарајућих хеуристика, по утицало на побољшање ефикасности рјешавача SAT и Max-SAT проблема базираног на *методи седиментације*. Утисак је аутора да би додавање већ постојећих хеуристика приближило рјешавач SAT и Max-SAT проблема базираног на *методи седиментације* са тренутно најбољим.
3. Имплементација у неком програмском језику, који се компајлира значајно, би повећала би ефикасност рјешавача.

5.5. ВАЈТХЕДОВ ПРОБЛЕМ МИНИМИЗАЦИЈЕ

Вајтхедов проблем минимизације је проблем проналажења ријечи минималне дужине у орбити аутоморфизама за дату ријеч. Проблем је од великог значаја у теорији група и топологији, и због тога је предметом интереса од 1936. године, када га је Вајтхед поставио (Whitehead, 1936).

Овдје је значајан као примјер који илуструје употребљивост *методе седиментације* на проблеме алгебре. У наставку подпоглавља уводе се потребне дефиниције да би у сљедећем подтоглављу 5.6, била изложена формулација WMP за рјешавање *методом седиментације,* без навођења експерименталних резултата. О самом WMP више у (Myasnikov & Haralic, 2006) и (Haralic, Miasnikov, & Myasnikov, 2005).

Дефиниција 5.5.1 *– Алфабет, инверзни алфабет, празна ријеч*

Нека је скуп $Y = \{y_1, \dots, y_n\}$ коначан *алфабет*; $Y^{-1} = \{y_1^{-1}, \dots, y_n^{-1}\}$ скуп *инверзног алфабета* Y и $Y^{\pm 1} = Y \cup Y^{-1}$. Подразумијева се да важи $(y^{-1})^{-1} = y$, за свако $y \in Y$. *Празну ријеч* обиљежавамо са ε.

Дефиниција 5.5.2 *– Редукована ријеч, правила редукције*

Ријеч $w = z_1 \dots z_m$ алфабета $Y^{\pm 1}$, зовемо *редукованом* ако важи услов:

$$(\forall i \in \{1, \dots, m-1\})\ z_i \neq z_{i+1}^{-1}. \tag{5.5.1}$$

Примјењујући правила редукције: $yy^{-1} \longrightarrow \varepsilon$ и $y^{-1}y \longrightarrow \varepsilon$ свака ријеч w алфабета може довести у редуковану форму $\bar{w}$. Редукована форма $\bar{w}$, је јединствена за сваку ријеч w, и не зависи од редослиједа редукција.

Дефиниција 5.5.3 *– Слободна група, ранг групе*

Означимо са $F = F(Y)$, *скуп свих редукованих ријечи* алфабета $Y^{\pm 1}$ и увеdимо операцију *множења* за $u, v \in F$, као $u \cdot v = \overline{uv}$. Структуру $(F, \cdot)$ називамо слободном групом са базом Y. Кардиналност скупа $|Y|$, зовемо рангом групе F, и обиљежавамо као $\mathrm{rang}(F)$. Такође, користимо краћу ознаку F_n за $\mathrm{rang}(F) = n$.

Дефиниција 5.5.4 *– Аутоморфизам, скуп аутоморфизама*

Бијекцију $\phi: F \to F$, зовемо *аутоморфизмом* ако важи услов:

$$(\forall u, v \in F)\ \phi(uv) = \phi(u)\phi(v). \tag{5.5.2}$$

Скуп свих аутоморфизама групе $(F,\cdot)$ означавамо са $\mathrm{Aut}(F)$.

Скуп свих аутоморфизама $\mathrm{Aut}(F)$ и операција слагања функција чине такође групу, коју означавамо са $(Aut(F),\circ)$. Сваки аутоморфизам ϕ потпуно је одређен сликама алфабета $\phi(y)$, за $y \in Y$.

Дефиниција 5.5.5 *– Нилсенов аутоморфизам*

Аутоморфизам $\phi \in \mathrm{Aut}(F)$ зовемо *Нилсеновим аутоморфизмом* ако за неко $z \in Y$ важи:

$$(\forall y \in F)\, z \neq y \;\Rightarrow\; \phi(y) = y \quad \text{и} \tag{5.5.3}$$

$$\phi(z) = z^{-1} \;\vee\; \phi(z) = y^{\pm 1}z \;\vee\; \phi(z) = zy^{\pm 1}\,, \text{ за неко } y \in Y, y \neq z. \tag{5.5.4}$$

Примјетимо да аутоморфизам за који $\phi(z) = z^{-1}$, а све остали елемете се сликају у саме себе не може смањити дужину ријечи. Такве аутоморфизме зовемо инваријантним за дужину ријечи. Скуп $N(Y)$ чине сви Нилсенови аутоморфизми, сем оних који су инваријантни за дужину ријечи.

Дефиниција 5.5.6 *– Вајтхедов аутоморфизам*

Нетривијалан аутоморфизам $\phi \in \mathrm{Aut}(F)$ зовемо *Вајтхедов аутоморфизмом* ако задовољава један од два услова:

1. ϕ пермутује $Y^{\pm 1}$ елементе или
2. ϕ слика у самог себе неки елемент $z \in Y^{\pm 1}$, тј. $\phi(z) = z$, а за остале елементе $y \in Y^{\pm 1}$, такве да $z \neq y^{\pm 1}$ важи:

$$\phi(y) = y \;\vee\; \phi(y) = yz \;\vee\; \phi(y) = z^{-1}y \;\vee\; \phi(y) = z^{-1}yz. \tag{5.5.5}$$

Непосредно се закључује да су Вајтхедови аутоморфизми који задовољавају услов 1. инваријантни за дужину ријечи. Означимо са $W(Y)$ скуп свих Вајтхедових аутоморфизама, који задовољавају услов 2.

Очигледно, сваки Нилсенов аутоморфизам је и Вајтхедов аутоморфизам, тј. $N(Y) \subseteq W(Y)$. Такође, непосредно закључујемо да важи:

$$|N(Y)| = 4n(n-1) \quad \text{и} \quad |W(Y)| = 2n(4^{n-1}-1), \tag{5.5.6}$$

гдје, $n = |Y|$.

Познато је да је сваки аутоморфизам $\phi \in \mathrm{Aut}(F)$ производ коначно много Нилсенових, а тиме и Вајтхедових аутоморфизама (Lyndon & Schupp, 1977).

Дефиниција 5.5.7 *– Орбита аутоморфизам*

Орбиту аутоморфизама за ријеч $w \in F$, дефинишемо као скуп свих аутоморфинис слика ријечи w:

$$\mathrm{Orb}(w) = \{v \in F \mid (\exists \phi \in Aut(F))\ \phi(w) = v\}. \tag{5.5.7}$$

Дефиниција 5.5.8 *– Минимална ријеч*

Ријеч $w \in F$ зовемо *минималном,* или *аутоморфно минималном*, ако важи:

$$(\forall \phi \in Aut(F))\ |w| \leq |\phi(w)|. \tag{5.5.8}$$

Ријеч минималне дужине у $\mathrm{Orb}(w)$ означавамо са $w_{\min}$. Ријеч $w_{\min}$ не мора бити јединствено одређена у $\mathrm{Orb}(w)$, односно може постојати више међусобно различитих ријечи минималне дужине.

Дефиниција 5.5.9 *– Вајтхедов проблем минимизације*

За дату ријеч $w \in F$ пронаћи аутоморфизам $\phi \in Aut(F)$, такав да $w_{\min} = \phi(w)$.

Рјешење Вајтхедовог проблема минимазиције (WMP) базира се на Вајтхедовој теореми доказаној у (Whitehead, 1936), а овдје је наведена без доказа.

Теорема 5.5.1 *– Вајтхедова теорема*

Нека је $w \in F_n(Y)$. Ако је $|w| \geq |w_{\min}|$, тада постоји Вајтхедова аутоморфизам $\phi \in W(Y)$, такав да $|w| \geq |\phi(w)|$.

Теорема 5.5.1 даје основу за алгоритам за рјешење WMP. Уведимо прво појам аутоморфизма који смањује дужину ријечи и формулишимо алгоритам.

Дефиниција 5.5.10 *– Аутоморфизам који смањује дужину ријечи*

Аутоморфизам $\phi \in Aut(F)$ називамо *аутоморфизмом који смањује дужину ријечи* $w \in F$, ако $|\phi(w)| < |w|$.

Алгоритам 5.5.1 *– Вајтхедов алгоритам*

Дата је ријеч $w \in F$, задатак алгоритма је да пронађе аутоморфизам $\phi \in Aut(F)$, такав да $w_{\min} = \phi(w)$.

Означимо са $w_1 = w$. На основу теореме 5.5.1, ако $w_1 \neq w_{\min}$, тада постоји Вајтхедов аутморфизам $\phi_1 \in W(Y)$, такав да $|w_1| \geq |\phi_1(w_1)|$, тј. који смањује дужину ријечи w_1. Овај услов је одлучив, јер је кардиналност Вајтхедових аутоморфизама коначна, $|W(Y)| = 2n(4^{n-1} - 1)$. У том случају означимо са $w_2 = \phi_1(w_1)$, и наставимо описани поступак, све док у неком кораку k не постоји Вајтхедов аутморфизам $\phi_k \in W(Y)$, такав да $|w_k| \geq |\phi_k(w_k)|$. Тада на основу теореме 5.5.1 слиједи да је $w_{\min} = w_k$ и да је $\phi = \phi_k \circ \phi_{k-1} \circ \dots \circ \phi_1$ аутоморфизам који пресликава w у $w_{\min}$.

На основу формулације алгоритма 4.2.1 закључујемо да важи сљедећи низ неједначина:

$$|w| = |w_1| > |w_2| > \cdots > |w_k| = |w_{\min}|. \tag{5.5.9}$$

Низ неједначина има за директну посљедицу то да ће Вајтхедов алгоритам 5.5.1 имати највише $|w|$ корака, што значи да је тотално коректан. Из ове чињенице у (Myasnikov & Haralic, 2006) може се наћи доказ оцјене комплексности $O\big(2cn|w|^2(4^{n-1} - 1)\big)$, гдје је $0 \leq c \leq 3$, затим $n = \text{rang}(F)$ и $|w|$ дужина улазне ријечи.

Наведене дефиниције и теореме послужиће нам као основа за моделовање WMP за рјешавање *методом седиментације*.

5.6. КОМБИНАТОРНА ФОРМУЛАЦИЈА ВАЈТХЕДОВОГ ПРОБЛЕМА МИНИМАЛИЗАЦИЈЕ ЗА МЕТОД СЕДИМЕНТАЦИЈЕ

Циљ моделовања је да моделујемо WMP, тако да добијемо Вајтхедов алгоритам 5.5.1 и на тај начин да покажемо општост *методе седиментације*.

На основу Вајтхедов алгоритама 5.5.1 за дату ријеч w, најдужи број корака алгоритма је $|w|$, стога узимамо да је $l = |w|$. Скуп промјенљивих одлучивања је, као и раније, $X = \{x_1, \dots, x_l\}$.

Скуп домена $Dom = \{D_1, \dots, D_l\}$ за промјенљиве одлучивања биће скуп Вајтхедових аутоморфизама, тј. $x_i \in D_i = W(Y)$, $i \in L$.

Будући да су вриједности промјенљивих одлучивања Вајтхедови аутоморфизми, означимо са $\chi_0(y) = y$, за $y \in Y$ и са $\chi_\theta = x_\theta \circ x_{\theta-1} \circ \ldots \circ x_1$, $i \in L$.

Скуп услова Φ је празан, јер аутоморфизми чијом композицијом долазимо до минималне ријечи, нијесу међусобно условљени. Услов изласка $F(\theta)$ биће $D_\theta \neq \emptyset$, тј. алгоритам ради све док има Вајтхедових аутоморфизама који могу смањити дужину ријечи w.

Функција критеријума може се једноставно формулисати као $f(x_1, \ldots, x_l) = |\chi_\theta(w)|$, гдје је θ, највеће за које $D_\theta \neq \emptyset$. Ова формулација није у складу са формулом (4.1.1). Будући да је превођење у облик дат формулом (4.1.1) техничке природе изостављамо га.

Скуп хеуристичких релација $\Xi = \{\prec_1, \ldots, \prec_l\}$, једноставно дефинишемо за свако $\prec_i$, $i \in L$, као затворење до релације тоталног поретка релације $\ll_i$, коју дефинишемо за $\phi_1, \phi_2 \in D_i$, као:

$$\phi_1 \ll_i \phi_2 \Leftrightarrow |\phi_1 \circ \chi_{i-1}| < |(\phi_2 \circ \chi_{i-1})(w)|. \tag{5.6.1}$$

Из формуле (5.6.2) слиједи да сматрамо да су бољи они Вајтхедови аутоморфизми из домена D_i, који заједно са претходно одређеним вриједностима промјенљивих одлучивања, више скраћују дужину ријечи w.

Процедуре и функције: FindSolution(θ, Dom), Estimation(θ, Dom), NonDetVarEstimation(θ), ReportSolution() и CutOff(θ, ε) су управо онакве какве су описане у потпоглављу 4.1.3. Функција Sediment(θ, x) брише све Вајтхедове аутоморфизме из $x \in D_{\theta+1}$ који не доводе до смањења дужине ријечи :

$$|\chi_\theta(w)| \leq |(x \circ \chi_\theta)(w)|. \tag{5.6.2}$$

Потребно је и довољно да се бришу такви хомоморфизми само из домена $D_{\theta+1}$. Остале скупове домена $D_{\theta+2}, \ldots D_l$ функција Sediment(θ, x) оставља неизмјењеним.

На крају остало је још моделовање функција Estimation(θ, Dom) и NonDetVarEstimation(θ). Примјетимо да је природа Вајтхедовог алгоритма 5.1.1 итеративна, оног тренутка када смо испунили услов изласка тј. ако смо пронашли оптимум, нема сврхе да испитујемо остале могућности, сем уколико не желимо да и нађемо сва оптимална рјешења, којих може

бити више. То нам није намјера, па функцију $\text{Estimation}(\theta, Dom)$ треба тако дефинисати да је она 0, док не дођемо до првог рјешења, а потом, било која вриједност већа или једнака од оптимума. Имајући у виду да је иницијална вриједност за оптимум $minimum = +\infty$, линија (**21**) псеудокода 4.1.2, наведена својства функције $\text{Estimation}(\theta, Dom)$ можемо остварити дефинисање функције $\text{Estimation}(\theta, Dom) = +\infty - \ minimum$. При томе сматрамо да важи $\infty - \infty = 0$. За овако дефинисану функцију $\text{Estimation}(\theta, Dom)$, функцију $\text{NonDetVarEstimation}(\theta)$ дефинишемо као увијек једнаку 0.

Из описаног моделовања WMP-а за *метод седиментације* непосредно изводимо да он задовољава све услове из главе 4, те на тај начин можемо направити рјешавач за WMP. Циљ овог примјера је да покаже општости и примјенљивост методе седиментације. Код појединих проблема примјена методе седиментације довешће до већ познатих метода, ако се моделовање ослања на претпоставкама које су исте, или сличне, већ постојећим методама. Такав је случај са WMP.

6. ПРИМЈЕНА МЕТОДА СЕДИМЕНТАЦИЈЕ НА ПРОБЛЕМЕ У ТРАНСПОРТУ

Проблеми оптимизације у транспорту представљају широку класу проблема, на којима су примјењивање методе оптимизације. У овом раду изабран је *проблем додјеле везова бродовима у контејнерским лукама*, у литератури на енглеском језику познат је под именом *Berth Allocation Problem,* скраћено BAP. Ради се цијелој класи проблема, чији се примјери могу разликовати по начину моделовања луке и бродова, броју везова који се додјељује бродовима, критеријумској функцији, итд. Због свог употребе коју има, BAP је један од најважнијих проблема лучког планирања. За разне његове варијанте постоје бројна рјешења, од оних на теоретском, до оних на апликативном нивоу. У наставку ћемо изложити примјену *методе седиментације* на BAP-у.

6.1. ПРОБЛЕМ ДОДЈЕЛЕ ВЕЗОВА

BAP се састоји у додјељивању веза броду, којег треба опслужити у контејнерској луци, током периода за који се планира распоређивање бродова у луци. За сваки брод имамо податке о његовом очекиваном времену упловљавања, времену испловљавања брода, величини брода, времену које је потребно за извршење лучких операција истовара и утовара, везу на којем везивање брода има најниже трошкове лучких операција, казне које плаћа лука уколико пожурује брод са доласком, касни са почетком лучких операција, задржавање иза времена када је брод требао да исплови. Неформално, BAP можемо дефинисати као *проблем додјељивања броја веза, у одређеном временском интервалу, свим бродовима за које планирамо лучке операције, тако да је функцију критеријума минимализујемо*. Доказано је у (Lim, 1998) да је BAP тешки NP проблем *(NP-hard)*.

6.1.1. BAP класификација

BAP се класификује као *дискретни*, *континуални* и *хибридни*. У случају дискретног BAP-а, на енглеском језику *Descrete Berth Allocation Problem* у ознаци DBAP, сваки вез у луци може да опслужује само један брод у једном тренутку, односно брод може да заузме, у току свог боравка у луци, само један вез. Вријеме је такође подијељено у дискретне јединице. Хибридни BAP, на енглеском језику *Hybrid Berth Allocation Problem* у ознаци HBAP, је сличан

као DBAP, са том разликом да брод у једном тренутку може да заузима више сусједних везова. Неки модели HBAP-а дозвољавају и то да мањи бродови дјеле вез. У континуалном BAP-у лука није подијељена на дискретне везеове, већ се дозвољава да брод може бити привезан на било којем мјесту (уколико то физички услови дозвољавају) дока. Континални BAP, на енглеском *Continuous Berth Allocation Problem* у ознаци CBAP, није заправо континуалан у математичком смислу. Док је подијељен на мање јединице дужине (*10 m*), што и њега чини дискретним, само је интерпретација дискретне јединице различита у односу на DBAP и HBAP. У случају CBAP-а дискретна јединица је дужина, док у случају DBAP-а и HBAP-а то је цијели простор на којем се веже један, или више бродова.

Разликујемо такође *динамички* и *статичку* верзију BAP-а. У статичкој верзији подразумијева се да вријеме доласка брода у луку није стриктно ограничење. Претпоставља се да брод већ чека у луци, или да он може доћи у луку прије времена када је најављено његово упловљење у луку. Пожуривање брода је могуће, али за то лука плаћа пенале (казне) бродарској компанији. У динамичкој верзији вријеме упловљавања брода у луку се не може мијењати, у односу на очекивано.

Детаљи о класификацији BAP-а могу се наћи у (Meisel, Seaside Operations Planning in Container Terminals, 2009) и (Bierwirth & Meisel, 2010).

6.1.2. BAP модел

Проблем додјеле везова бродовима у луци је комплексан. Састоји се од планирања везова *(Berth Planning Problem),* проблема додјеле везова *(Berth Allocation Problem),* проблема додјеле лучких кранова *(Quay Crane Assignment Problem)* и проблема распоређивања лучких кранова (*Quay Crane Scheduling Problem*). Сви побројани проблеми могу се рјешавати одвојено као у (Cordeau, Laporte, Legato, & Moccia, 2005), (Imai, Nishimura, Hattori, & Papadimitriou, 2007), (Hansen, Oguz, & Mladenović, 2008), (Fu & Diabat, 2015) или заједно (Meisel & Bierwirth, A framework for integrated berth allocation and crane planning in seaport container terminals, 2013), (Vacca, Salani, & Bierlaire, 2011).

За илустрацију рада *методе седиментације* одабране су варијанте DBAP и HBAP, које минимизују казне (пенале) луке. У том циљу коришћен је *Рашиди-Цанī* модел уведен у (Rashidi & Tsang, 2013). Коришћен је само дио

модела који се односи на BAP, и њега наводимо онако како је он формулисан у (Rashidi & Tsang, 2013). Дио модела, који се односи на додјелу кранова је изостављен. Подразумијева се да сваки вез располаже са тачно једним краном.

Метод седиментације користи само улазне податке и функцију критеријума Рашиди-Цанг модела. Остали елементи, укључујући промјенљиве одлучивања и ограничења, не користе се онако како је то формулисано у Рашиди-Цанг моделу. Умјесто тога, промјенљиве одлучивања су имплементиране у виду функција, јер се користе за израчунавање функције критеријума. Ограничења су имплеметирана у конструкцији рјешења, ажурирањем простора рјешења у којем се врши претрага за оптималним рјешењем.

6.1.3. Претпоставке модела

Претпостављамо да важе сљедеће претпоставке у вези додјеле везова бродовима у контејнерским лукама.

Претпоставка 6.1.1 *– О времену проведеном у лици*

Сваки брод има унапријед одређено вријеме боравка у луци. Лука плаћа казне (пенале) уколико, раније или касније додјели вез броду од предвиђеног, такође казне (пенали) се плаћају ако брод касније напусти луку од предвиђеног времена испловљавања.

Претпоставка 6.1.2 *– О везу са најмањим трошковима*

Сваки брод ће имати у луци вез, на којем су трошкови истовара и утовара најмањи. Овај вез обично је одређен близином магазина у којем се смјешта терет истог типа, којег утоварује или истоварује брод. Везивањем брода даље од таквог веза, подразумијева додатне трошкове транспорта, па због тога лука плаћа казне (пенале).

Претпоставка 6.1.3 *– О доступности бар једног крана за утовар и истовар*

У свакој временској јединици, одређен број контејнера, договором узимамо да је то један, може бити утоварен или истоварен на везани брод. Штовише, подразумијева се да на везу увијек доступан један кран за операције утовара и истовара брода. Ова претпоставка нам је битна ради успостављања везе са Рашиди-Цанг моделом, који у општем случају дозвољава и доступност већег броја кранова.

Наведене претпоставке су стандардне за DBAP и HBAP.

6.1.4. Улазни подаци

Модел и алгоритам подразумјева сљедеће улазне податке:

T : Укупан број временских јединица, за који се планира додјела веза. Број временски интервала називамо *хоризонт планирања*.

m: Укупан број везова у луци.

l: Укупан број бродова којима треба додијелити вез, током хоризонта планирања.

$vessel$: Низ са подацима за планирање додјеле веза за сваки брод. Можемо га представити као низ:

$$vessel = \langle (ETA_k, a_k, b_k, d_k, s_k, C_{1k}, C_{2k}, C_{3k}, C_{4k}) \mid k \in L \rangle \quad (6.1.1)$$

При томе, елементи 9-торке представљају сљедеће податке:

ETA_k : Вријеме очекиваног доласка брода $vessel_k$.

a_k : Вријеме, у временским јединицама, потребно за обраду утовара и претовара у луци брода $vessel_k$.

b_k : Број везова, које заузима брод $vessel_k$.

d_k : Вријеме када брод $vessel_k$ треба да исплови.

s_k : Индекс најјефтинијег *(„омиљеног“)* веза брода $vessel_k$.

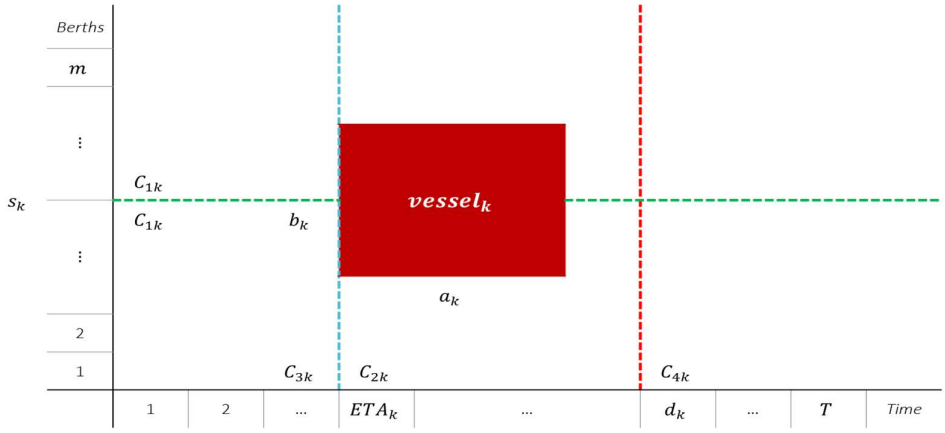
C_{1k} : Цијена казне, уколико брод $vessel_k$, није за један вез привезан у односу на његов најјефтинији вез s_k.

C_{2k} : Цијена казне по временској јединици уколико брод $vessel_k$ мора да буде привезан прије очекиваног времена његовог доласка ETA_k.

C_{3k} : Цијена казне по временској јединици уколико брод $vessel_k$ мора да буде привезан након очекиваног времена његовог доласка ETA_k.

C_{4k} : Цијена казне по временској јединици уколико брод $vessel_k$ мора да остане дуже у везу од времена његовог испловљавања d_k.

У наставку ће бити приказано рјешавање DBAP-а и HBAP-а *методом седиментације*. Вриједност параметра b_k у случају DBAP може бити само 1, док ће вриједност истог параметра у случају HBAP бити између 1 и 3.
Улазни подаци су визуелно приказани на илустрацији 6.1.1.



Илустрација 6.1.1 *– Улазне промјенљиве за BAP*. На илустрацији је представљен брод, као црвени правоугаоник у матрици временских јединица и индекса бројева везова у луци.

6.1.5. Промјенљиве одлучивања

Наводимо даље промјенљиве одлучивања Рашиди-Цанг модела. Иако их *методом седиментације* не користи у облику који слиједи, наводимо их ради успостављања везе са изворним моделом:

At_k : Индекс временске јединице у којем ће броду $vessel_k$ бити додијељен вез.

Dt_k : Индекс временске јединице у којем ће брод $vessel_k$ напустити вез.

Bi_k : Вриједност најмањег индекса веза који је додијељен броду $vessel_k$.

X_{itk} : Бинарна промјенљива $X_{itk} \in \{0,1\}$, која узима вриједност 1, ако је брод $vessel_k$ на везу i у временској јединици t, у сопротном је 0.

Промјенљиве одлучивања су визуелно приказане на илустрацији 7.1.2.

Berths										
m	0	0	0	0	⋯	0	0	0	0	
⋮	⋮	⋮	⋯	⋮	⋯	⋮	⋯	⋮	⋮	
$Bi_k + b_k - 1$	0	0	0	1	⋯	1	0	0	0	
⋮	⋮	⋮	⋮	⋮	⋯	⋮	⋮	⋮	⋮	
Bi_k	0	0	0	1	⋯	1	0	0	0	
⋮	⋮	⋮	⋯	⋮	⋯	⋮	⋯	⋮	⋮	
2	0	0	0	0	⋯	0	0	0	0	
1	0	0	0	0	⋯	0	0	0	0	
X_{itk}	1	2	...	At_k	...	$At_k + a_k - 1$	Dt_k	...	T	Time

Илустрација 6.1.2 *– Промјенљиве одлучивања BAP*. На илустрацији је представљен брод, као црвени правоугаоник са означеним вриједностима промјенљивих одлучивања у матрици временских јединица и индекса бројева везова у луци.

6.1.6. Ограничења

Допустиво рјешење BAP-а мора да задовољава ограничења, која даље наводимо.

Ограничење 6.1.1 *– На везу је само један брод у датом тренутку*

У тренутку t, сваки вез је додијељен највише једном броду:

$$(\forall i \in \{1, \dots, m\})(\forall t \in \{1, \dots, T\}) \sum_{k=1}^{l} X_{itk} \leq 1. \tag{6.1.2}$$

Ограничење 6.1.2 *– На везу је додијељен броду само у току веза*

Вез је додијељен броду само у времену од његовог везивања до његовог испловљавања:

$$\begin{aligned}&(\forall k \in \{1, \dots, l\})(\forall t \in \{1, \dots, T\})(\forall i \in \{1, \dots, m\})\\ &\quad (At_k \leq t \leq Dt_k \ \wedge \ Bi_k \leq i < Bi_k + b_k \implies X_{itk} = 1) \wedge\\ &\quad (t < At_k \ \vee \ Dt_k < t \ \vee \ i < Bi_k \ \vee \ Bi_k + b_k \leq i \implies X_{itk} = 0)\end{aligned} \tag{6.1.3}$$

6.1.7. Функција критеријума

Уведимо прво помоћни низ Z_k, која представља суму апсолутних вриједности разлика између најјефтинијег веза s_k и везова додијељених броду $vessel_k$, за свако $k \in \{1, \dots, l\}$, као:

$$Z_k = \sum_{t=1}^{T} \sum_{i=1}^{m} \begin{cases} |i - s_k| & : \ X_{itk} = 1, \\ 0 & : \ X_{itk} = 0. \end{cases} \tag{6.1.4}$$

Тада функцију критеријума, коју треба минимализовати, можемо формулисати као:

$$\begin{aligned}VesselCost = \sum_{k=1}^{l} \{&C_{1k} Z_k + C_{2k}(ETA_k - At_k)^+\\ &+ C_{3k}(At_k - ETA_k)^+ + C_{4k}(Dt_k - d_k)^+\}.\end{aligned} \tag{6.1.5}$$

У формули (7.1.5), користи се оператор a^+, који се дефинише као:

$$a^+ = \begin{cases} a & : \ a > 0, \\ 0 & : \ a \leq 0. \end{cases} \tag{6.1.6}$$

Први производ $C_{1k}Z_k$, у члану суме у формули (7.1.5), изражава казну због додјеле везова удаљених од најјефтинијег веза за брод $vessel_k$. Други производ $C_{2k}(ETA_k - At_k)^+$ изражава казну за пожуривање брода $vessel_k$, тј. његову додјелу веза прије него што је била очикавина ETA_k. Трећи производ $C_{3k}(At_k - ETA_k)^+$ изражава казну за кашњење у додјели веза броду $vessel_k$, додјели веза након очекиваног доласка ETA_k. Коначно, четврти производ $C_{4k}(Dt_k - d_k)^+$ изражава казну за кашњење испловљавања брода $vessel_k$, тј. одласка након d_k.

Функција критеријума минимализује казну мјеста веза, чекања брода, пожуривања брода и кашњења брода. Према Мејселовој класификацији BAP-а, датој у (Meisel, Seaside Operations Planning in Container Terminals, 2009) и (Bierwirth & Meisel, 2010), Рашиди-Цанг модел се класификује као:

$$disc \text{ or } hybrid \mid stat \mid fix \mid \Sigma(w_1\ wait + w_2\ speed + w_3\ tard + w_4\ pos).$$

Изворна формулација функције критеријума у Рашиди-Цанг моделу (Rashidi & Tsang, 2013), није сагласна са *„performance measure”* Мејселовој класификацији. У (Rashidi & Tsang, 2013) редослијед је: $\Sigma(w_4\ pos + w_1\ wait + w_2\ speed + w_3\ tard)$.

6.1.8. Неколико примједби о Рашиди-Цанг BAP моделу

Рашиди-Цанг намјењен је за моделовање дискретног, континуалног и хибридног облика BAP. Математички модел је исти за све три форме, само се мијења интерпретације јединица везова и јединице времена. Прегледна табела са интерпретацијама за ове три форме је дата у табели 6.1.1.

Из табеле 6.1.1 је јасно да CBAP није континуалан, у уобичајеном значењу тог термина у математици. Он је такође дискретан, само је интерпретација јединице веза другачија.

У случају DBAP-а и HBAP-а сматра се да везови имају исту физичку дужину, иако у стварана дужине не мора у потпуности бити иста. У случају HBAP-а, за бродове који заузимају више од једног веза, увијек ће постојати трошак транспорта терета до најјефтинијег веза, чак ако је најјефтинији вез додијељен броду. То је посљедица дефиниције помоћног низа Z_k, датог у формули (6.1.4). Ова појава у Рашиди-Цанг моделу BAP се оправдава

постојањем трошкова пребацивања терета са сусједних везова, додијељених броду, до најјефтинијег веза.

***Табела 6.1.1** – Интерпретација јединица везова и јединице времена у Рашиди-Цанг моделу*

Облик BAP	*Интерпретација јединице веза*	*Интерпретација јединице времена*
Дискретни DBAP	Док је подијељен на дјелове које називамо *везовима*. Само један брод може бити опслужен на једном везу, односно вез је додијељен само једном броду у једној временској јединици. Због тога је b_k=1 у случају DBAP-а.	Јединица времена представља обично временски интервал од неколико сати. У тест инстанцама временска јединица је период од 3 сата. Како је хоризонт планирања 1 или 2 седмице, то значи је број временских јединца 56 или 112. Оваква интерпретација јединица се користи и у (Giallombardo, Moccia, Salani, & Vacca, 2010)
Континуални CBAP	Док је подијељен на јединице извјесне дужине, изражене у метрима. У (Meisel, Seaside Operations Planning in Container Terminals, 2009) и (Bierwirth & Meisel, 2010) дужина мјере је 10 метара. За сваки брод се израчунава његова дужина, изражена у јединици мјере од 10 метара.	Временска јединица је обично 1 сат. Та мјера се такође користи у (Meisel, Seaside Operations Planning in Container Terminals, 2009) и (Bierwirth & Meisel, 2010).
Хибридни HBAP	Као и у случају DBAP-а док је подијељен на дјелове које називамо *везовима*. Већи бродови могу да заузимају више везова. У тест инстанцама велики бродови могу да заузимају до 3 веза. Неки модели HBAP-а дозвољавају да мањи бродови могу да дјеле вез. То није случај са Рашиди-Цанг моделом.	Такође, као и у случају DBAP-а временска јединица је период од 3 сата. Хоризонт планирања је такође једна или двије седмице.

6.2. ПРЕГЛЕД ПОСТОЈЕЋИХ МЕТОДА ЗА РЈЕШАВАЊЕ ПРОБЛЕМА ДОДЈЕЛЕ ВЕЗОВА

У литератури, нарочито оној новијој, има више приступа рјешавању BAP-а хеуристичким и мета-хеуристичким методама у односу на приступу који тачно рјешавају BAP. Према скорашњим прегледним радовима (Bierwirth & Meisel, 2010) и (Bierwirth & Meisel, 2015) хеуристичких и мета-хеуристичких приступа има око 75%, док приступа тачном рјешавању BAP има око 25%.

Будући да је *метод седиментације,* метод тачног рјешавања посебно ћемо се осврнути на неке приступе тачном рјешавању BAP-а.

Тачан приступ рјешавању BAP-а може се наћи у (Vacca, Salani, & Bierlaire, 2011). Аутори разматрају верзију BAP-а, назива *„тактички проблем додјеле везова бродовима“*, на енглеском језику *Tactical Berth Allocation Problem*

(TBAP), којег су увели (Giallombardo, Moccia, Salani, & Vacca, 2010). Он поред тога што додјељује везове бродовима прави и распоред додјеле кранова. TBAP је карактеристичан је по томе што разматра унапријед утврђене планове (распореде) кранова броду, који се називају профили. У (Vacca, Salani, & Bierlaire, 2011) се предлаже модел заснован на експоненцијалном броју промјенљивих, које се рјешавају методом генерисања колона, на енглеском језику *Column Generation*. Затим је имплементиран тачан *„гранај и наплати“* (*„Branch-and-Price“*) алгоритам који долази до оптималног рјешења. У раду су представљене неколике хеуристике, за главни проблем, као и за подпроблеме, који се могу употријебити и за друге методе, базиране на *„гранај и наплати“* алгоритму.

Рад (Hendriks, Armbruster, Lefeber, & Udding, 2012) разматра проблем BAP-а са становишта луке које има више терминала на којима се врши утовар и истовар контејнера. Посебно се разматрају проблем цикличног опслуживања бродова у овакве луке. Циљ је да балансирање оптерећења кранова, као и минимизовање транспортних трошкова у оквиру луке. У том циљу развијен је алгоритам базирам на мјешовитом цјелобројном линеарном програмирању.

Нови метод, под називом „*Combinatorial benders' cut algorithm“* развили су (Chen, Lee, & Cao, 2012) за рјешавање BAP-а, као и других лучких проблема. У овом приступу сви лучки проблеми третирају се заједнички и показује се да је такав приступ ефикаснији од *„гранај и сијеци“* приступа (*„Branch-and-Cut“)* приступа који је уграђен у CPLEX-у.

Проблемом додјеле везова и кранова (скраћено на енглеском језику BACAP), са становишта потрошње горива и емисијом штетних гасова са бродова разматрали су (Hu, Hu, & Du, 2014). У овом раду развијен је прво нови нелинеарни више критеријумски модел мјешовити цјелобројни цјелобројни модел који узима у обзир потрошњу и емисију штетних гасова бродова у луци, а затим се он трансформише у проблем мјешовитог цјелобројног конусног програмирања другог реда и као такав рјешава.

Иако је *проблем додјеле везова у лукама за расути терет,* различит у односу на проблем додјеле везова у контејнерским лукама, којим се ми бавимо, наводима и два рада који се баве овим проблемом. У (Umang, Bierlaire, & Vacca, 2013) проблем додјеле везова у лукама са расутим теретом предлаже

се *мијешани цјелобројни линеарни модел (Mixed Integer Linear Model),* који сагледава проблем са аспекта интеракције између проблема одлучивања који настају при додјели веза броду. Други приступ истом проблему налазимо у (Robenek, Umang, Bierlaire, & Ropke, 2014). У њему је циљ смањити вријеме опслуживања брода у луци. Њихов алгоритам се базира на *„ɩранај и наплати“* алгоритму. Главни проблем је представљен као проблем проналажења партиције скупа, односно подпроблема ради утврђивања колона са негативним редукцијом трошкова, који се рјешавају методама мјешовитог цјелобројног линеарног програмирања. За добијање допустивих рјешења користе мета хеуристички приступ базиран на методи *Претраживању околина (Neighborhood Search)*. Предложени алгоритам је тестиран на подацима добијеним из стварних лука расутог терета. Резултати показују да овај приступ може да ријеши проблеме у разумљивом времену за највише 40 бродова.

Као што је већ речено хеуристички и мета хеуристички приступи у рјешавању BAP доминирају. У наставку слиједи краћи прегед оваквих приступа само за приступе који рјешавају DBAP и HBAP.

Статички и динамички BAP рјешаван је у (Imai, Nishimura, & Papadimitriou, The dynamic berth allocation problem for container port, 2001). За обије варијанте проблема, додјела везова и низање бродова је утврђено тако да се минимизује вријеме чекања брода. Хеуристика базирана на *Лаɩранжовој релаксацији (Lagrangian Relaxation)* користи се за рјешавање проблема. Сличан приступ, са јачом Лагранжовом релаксацијом примијењен је у (Monaco & Samara, 2009) за динамичку верзију BAP-а.

(Cordeau, Laporte, Legato, & Moccia, 2005) су моделовали DBAP као *Multi-Depot Vehicle Routing Problem* са временским прозорима и примијенили мета хеуристику *Табу ɩретраживања (Tabu Search)* за проналажење добрих субоптималних рјешења проблема. Сличан приступ рјешавању BAP, користећи моделовање BAP као „*Multi-Depot Vehicle Routing Problem”,* имају и (Mauri, Oliveira, & Lorena, 2008), који користећи *алɩоритам „обуке ɩопулације“ (Population Training Algorithm)* комбинован са линеарним програмирањем *ɩенеришу колоне (Column Generation)* како би добили резултат.

Мета хеуристика *Симулирано каљење (Simulated Annealing)* коришћена је у више приступа рјешавању BAP-а. (Zhen, Lee, & Chew, 2011) разматра BAP у

условима несигурних времена упловљавања брова у луку, тако да се минимализује казне, које плаћа лука због одступања од првобитног плана додјеле везова. Предлажу модел и процедуру одлучивања у двије фазе, као и мета хеуристички приступ, базиран на *Симулиранм каљењу*, који је у стању да рјешава стварне проблеме са великим бројем бродова. Рад (de Oliveira, Mauri, & Lorena, 2012) посвећен је рјешавању динамичког облика DBAP-а, базира се на употреби *претраживања кластера (Clustering Search)* примјеном методе *Симулираног каљења* за генерисање рјешења.

Метода промјењивих околина (Variable Neighbourhood Search) користе (Hansen, Oguz, & Mladenović, 2008) и упоређују са другим мета хеуристичким приступима рјешавању BAP-а.

Генетски алгоритми су веома честа мета хеуристичка техника за рјешавање DBAP-а. За више варијанти DBAP-а Генетски алгоритме су примјењивали (Imai, Nishimura, & Papadimitriou, Berthing ships at a multi-user container terminal with a limited quay capacity, 2008) за минимизацију времена опслуживања брода у луци, за оне бродове, који су већ прекорачили временску границу опслуживања и привезани су спољне терминале. Нелинеаран модел распоређивања бродова, који је ријешен комбиновањем метода *Генетских алгоритама* и *Симулираног каљења* налазимо у (Han & Sun, 2006). Динамички стохастички BAP је разматран у (Zhou, Kang, & Lin, 2006), за којег је развијен *генетски алгоритам*. Један приступ заснован на методи *Генетских алгоритама* може се наћи у (Nishimura, Imai, & Papadimitriou, 2001).

Итеративну грамзиву методу за рјешавање DBAP-а користили су (Lin, Ying, & Wan, 2014) за оптимизацију укупног времена опслуживања брода.

HBAP са непромјењивим временима опслуживања бродова, мјешовито цјелобројно моделован, рјешаван је у (Chen & Hsieh, 1998). Графовску репрезентацију HBAP-а користили су (Moorthy & Teo, 2006). Приступ базиран на методи *Симулираног каљења* за рјешавање HBAP-а налазимо у (Dai, Lin, Moorthy, & Teo, 2008). *Оптимизација колонијом пчела,* коришћена је за рјешавање HBAP-а са непромјењивим временима опслуживања бродова, у (Kovač, Bee Colony Optimization Algorithm for the Minimum Cost Berth Allocation Problem, 2013), са становишта минимизације трошкова. Такође, у (Davidović, Kovač, & Stanimirović, VNS-based approach to minimum cost

hybrid berth allocation problem, 2015) разматра се проблем HBAP-а, са становишта минимизације трошкова. Користећи 3 типа околина за детерминистичку варијанту *методе промјењивих околина*, која се зове *метод промјењивог спуста*, покушава се наћи бољи вез за бродове који нијесу везани за најјефтиније мјесто веза.

HBAP формулације са временима опслуживања брода зависним од позиције додијељеног веза су предметом изучавања више радова. У (Imai, Nishimura, Hattori, & Papadimitriou, 2007) разматра се HBAP са увученим везовима. Такође, у (Imai, Nishimura, & Papadimitriou, Marine container terminal configurations for efficient handling of mega-containerships, 2013) развијен је *генетски алгоритам* за HBAP мега бродова, који се опслужују са двије стране. Генетски алгоритам је дизајниран и за верзију HBAP-а, која узима у обзир газ брода у (Nishimura, Imai, & Papadimitriou, 2001). Свођење HBAP-а на DBAP примјењују (Cordeau, Laporte, Legato, & Moccia, 2005).

Из овог кратког и непотпуног прегледа радова који се баве DBAP-м и HBAP-м довољан је за приказ варијанти проблема и мноштво метода, које се користе за њихово рјешавање.

6.3. КОМБИНАТОРНА ФОРМУЛАЦИЈА ПРОБЛЕМА ДОДЈЕЛЕ ВЕЗОВА ЗА МЕТОД СЕДИМЕНТАЦИЈЕ

Да би ријешили BAP *методом седиментације* тј. алгоритмима SEDA и SEDA+ERH, потребно је дефинисати све скупове, функције и релације из потпоглавља 4.1.1—3 тако да они моделују BAP. Уколико желимо да примијенимо и D&C, потребно је још дефинисати и релацију не-конфликности из дефиниције 4.3.2. Све наведене дефиниције скупова, функција и релација морају да задовољавају услове, који за њих треба да важе, како би били сигурни да такво моделовање резултује коректним алгоритмом за рјешавање BAP базираном на *методои-седиментације.*

6.3.1. Моделовање BAP за метод седиментације

У овом подпоглављу повезаћемо Рашиди-Цанг BAP модел са *методом седиментације*, краће ово повезивање ћемо називати *моделовање BAP-а за метод седиментације,* или још краће *моделовање BAP-а*.

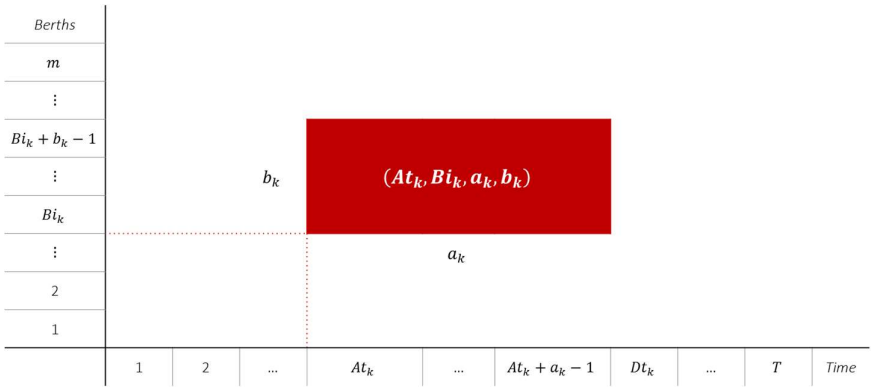
Подразумијевамо прво да су улазни подаци за BAP, дефинисани у потпоглављу 6.1.4, такође и улазни подаци за моделовање BAP-а.

Промјењиве одлучивања у Рашиди-Цанг BAP моделу су: At_k, Dt_k, Bi_k и X_{itk}, $k \in L$. Њихове дефиниције и значења наведена су потпоглављу 6.1.5. Из илустрације 6.1.2 јасно се види да је довољно да одредимо промјењиву одлучивања X_{itk}, $k \in L$ и да на основу ње одредимо остале промјењиве одлучивања. Међутим, за *метод седиментације*, боље је изабрати одређивање промјењивих одлучивања At_k и Bi_k, па на основу њихових вриједности израчунати Dt_k и X_{itk}, $k \in L$ користећи формуле:

$$Dt_k = At_k + a_k; \tag{6.3.1}$$

$$X_{itk} = \begin{cases} 1 & : \quad At_k \le t < At_k + a_k \ \wedge \ Bi_k \le i < Bi_k + b_k, \\ 0 & : \quad \text{иначе.} \end{cases} \tag{6.3.2}$$

Због тога довољно је одредити вриједност уређеног пара (At_k, Bi_k), да би на јединствен начин представили позицију брода $vessel_k$, $k \in L$ у матрици вријеме-вез. Уређени пар (At_k, Bi_k) зваћемо *позицијом брода* $vessel_k$, $k \in L$.



Илустрација 6.3.1 *– Промјењиве одлучивања BAP-а и њихова веза са позицијом брода.*
На илустрацији је представљен брод, као црвени правоугаоник са означеним вриједностима позиције брода (At_k, Bi_k) за $vessel_k$, $k \in L$ и начин на који из позиције израчунавамо остале промјењиве одлучивања BAP-а.

На илустрацији 6.3.1 илустрована је веза између позиције брода и осталих промјењивих одлучивања BAP-а. На основу наведеног можемо дефинисати скуп домена $Dom = \{D_1, \dots, D_l\}$. Пошто брод $vessel_k$ можемо представити као правоугаоник димензије $a_k \times b_k$ у матрици вријеме-вез димензије $T \times m$, домен D_k, $k \in L$ можемо одредити као:

$$D_k = \{(t, b) \mid t \in \{1, \dots, T - a_k + 1\} \ \wedge \ b \in \{1, \dots, m - b_k + 1\}\}. \tag{6.3.3}$$

Сваки брод повезујемо са једном промјењивом одлучивања, па ће промјењиве одлучивања у моделовању BAP-а за метод седиментације бити

скуп $X = \{x_1, \dots, x_l\}$. Метод седиментације одређује вриједност за сваку своју промјењиву одлучивања у одговарајућем скупу домена:

$$x_k = (\tau, \beta) = (At_k, Bi_k) \in D_k, \quad k \in L. \tag{6.3.4}$$

Дефинишимо даље функцију критеријума, тако што ћемо прво дефинисати f_k, $k \in L$ на сљедећи начин:

$$f_k\big((At_k, Bi_k)\big) = C_{1k}Z_k + C_{2k}(ETA_k - At_k)^+ + \\ +C_{3k}(At_k - ETA_k)^+ + C_{3k}(Dt_k - d_k)^+. \tag{6.3.5}$$

Помоћни низ Z_k, $k \in L$ дефинисан је формулом (6.1.4). Функцију критеријума f дефинишемо онда као:

$$f(x_1, \dots, x_l) = f\big((At_1, Bi_1), \dots, (At_l, Bi_l)\big) = \\ = \sum_{k=1}^{l} f_k\big((At_k, Bi_k)\big) = \sum_{k=1}^{l} f_k(x_k)\,. \tag{6.3.6}$$

Скуп хеуристичких релација $\Xi = \{\prec_1, \dots, \prec_l\}$, тако што дефинишемо понаособ сваку од релација $\prec_k$, $k \in L$, као затворење од потпуног поретка одговарајућих релација $\ll_k$, $k \in L$, дефинисаних као:

$$(\tau_1, \beta_1) \ll_k (\tau_2, \beta_2) \iff \\ f_k\big((\tau_1, \beta_1)\big) < f_k\big((\tau_2, \beta_2)\big) \vee \\ \big(\, f_k\big((\tau_1, \beta_1)\big) = f_k\big((\tau_2, \beta_2)\big) \wedge |\beta_1 - s_k| < |\beta_2 - s_k| \,\big). \tag{6.3.7}$$

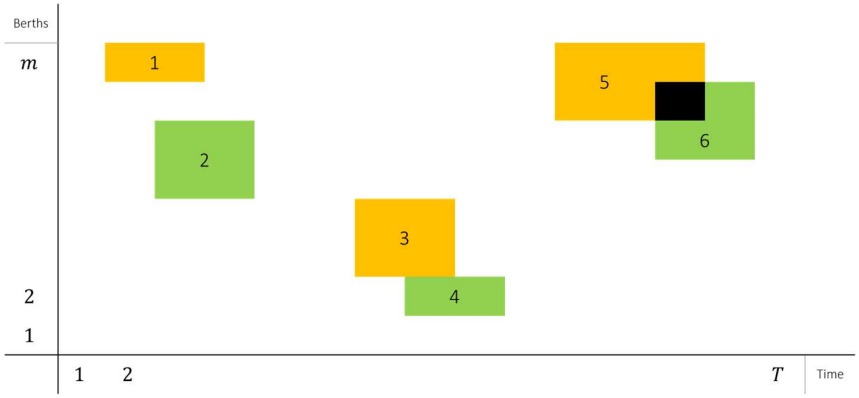
Функцију $F(\theta)$, дефинишемо као увијек нетачну. У конструкцији допустивог рјешења алгоритам ће одређивати све вриједности за промјењиве одлучивања.

Релацију неконфликтности ρ за двије позиције дефинишемо $(\tau_i, \beta_i) \in D_i$ и $(\tau_j, \beta_j) \in D_j$, за $i, j \in L$ дефинишемо формулом:

$$(\tau_i, \beta_i)\ \rho\ (\tau_j, \beta_j) \iff \\ \tau_i + a_i \le \tau_j \vee \tau_j + a_j \le \tau_i \vee \\ \beta_i + b_i \le \beta_j \vee \beta_j + b_j \le \beta_i. \tag{6.3.8}$$

Релација ρ је релација не преклапања два правоугаоника, чије су ивице паралелне са координатним осама. Координате доњег лијевог тјемена првог

правоугаоника је (τ_i, β_i), а координата доњег лијевог тјемена другог правоугаоника (τ_j, β_j). Ширина првог правоугаоника је a_i, а другог a_j. Висина првог правоугаоника је b_i, а другог b_j. При томе вриједности за a_i, b_i, a_j и b_j, за неко $i, j \in L$, одређене су потпоглављу 6.1.4. Релацију конфликтности $\bar{\rho}$, дефинишемо као негацију релације неконфликтности ρ.



Илустрација 6.3.2 *– Релација неконфликтности ρ и релација конфликтност $\bar{\rho}$.* Бродови 1 и 2, као и 3 и 4 су у релацији не-конфикнтонсти ρ, али нијесу у релацији конфликтности $\bar{\rho}$. Бродови 5 и 6 нијесу у релацији неконфликтности ρ, али јесу у релацији конфликтности $\bar{\rho}$.

Процедуре и функције: FindSolution(θ, Dom), Estimation(θ, Dom), NonDetVarEstimation(θ), ReportSolution() и CutOff(θ, ε) су управо онакве какве су описане у потпоглављу 4.1.3. Функција Sediment(θ, x) брише све позиције у доменима неодређених промјењивих одлучивања које су у конфликту са додјелом позиције $x_\theta = x$.

Из описаног моделовања BAP-а за *метод седиментације* непосредно изводимо да он задовољава све услове из поглавља 4.1. То за посљедицу има да комбиновањем овог моделовања BAP-а са алгоритмима SEDA, SEDA+ERH и SEDA+ERH+D&C добијамо три рјешавача *(solvera)* за BAP. Њихове карактеристике биће предметом пажње сљедећег подпоглавља.

6.3.2. Комплексност SEDA и SEDA+ERH за рјешавање BAP-а

Након моделовања BAP-а за *метод седиментације*, описаног у претходном подпоглављу у прилици смо да прецизније дамо оцјену комплексности алгоритама SEDA и SEDA+ERH за рјешавање BAP-а. У наставку оцјене комплексности концентрисаћемо се на случај DBAP-а. Веома слична процјена се може направити и за HBAP.

Без губитка општости претпоставићемо да је вријеме обраде брода 1 временска јединица, као и да су казне за не везивање брода за најјефтинији вез, пожуривање брода и чекање на додјелу веза 1, док је казна за кашњење испловљавања брода из луке 0. Наведене претпоставке можемо сумирати формулом:

$$(\forall k \in L)\ a_k = 1 \wedge C_{1k} = 1 \wedge C_{2k} = 1 \wedge C_{3k} = 1 \wedge C_{4k} = 0. \tag{6.3.9}$$

Управо описани облик улазних података представља најгори сценарио *(worst case scenario)* за улазне податке DBAP-а. За такав сценарио функција f_k, за $k \in L$ приказана је на илустрацији 6.3.3.

Berths											
m											
⋮						⋮					
$s_k + 2$				4	3	2	3	4			
$s_k + 1$				3	2	1	2	3			
s_k			…	2	1	0	1	2	…		
$s_k - 1$				3	2	1	2	3			
$s_k - 2$				4	3	2	3	4			
⋮						⋮					
2											
1											
$\boldsymbol{f_k}$	1	2	…	$ETA_k - 2$	$ETA_k - 1$	ETA_k	$ETA_k + 1$	$ETA_k - 2$	…	T	Time

Илустрација 6.3.3 *– Приказ функције f_k у најгорем сценарију улазних података за DBAP.*

Из илустрације 6.3.3 непосредно закључујемо да постоји једна позиција (ETA_k, s_k) са казном 0, тј. $f_k\big((ETA_k, s_k)\big) = 0$. Такође, непосредно закључујемо да има $4i$ позиције са казном i, тј. вриједношћу функције f_k једнаке i, за $i > 1$.

Означимо са M неко горње ограничење вриједности оптимума, које ћемо звати *процјеном оптимума*. Комплексност SEDA-а за рјешавање BAP-а може се изразити бројем листова (или еквивалентно максималних грана) у стаблу претраживања простора рјешења, односно рекурзивном стаблу функције $\mathrm{FindSolution}(\theta, Dom)$. Број листова у стаблу претраживања простора рјешења за l бродова и процјену M означавамо са $\Psi(l, M)$. Функција $\Psi(l, M)$ може се рекурзивно израчунати на сљедећи начин:

$$\Psi(0,M)=1,\quad \Psi(l,0)=1 \quad \text{и}$$
$$\Psi(l,M)=\Psi(l-1,M)+4\sum_{i=1}^{M} i\Psi(l-1,M-i). \tag{6.3.10}$$

Лист у стаблу претраживања простора рјешења се досеже, ако смо одредили позиције свим бродовима, тј. $l=0$, па је зато $\Psi(0,M)=1$. Такође, ако је процјена оптималног рјешења $M=0$, тада је претрага у стаблу претраживања простора рјешења завршена, јер се рјешење састоји само од позиција са казном 0, које се налазе у јединој грани стабла. Због тога $\Psi(l,0)=1$.

Број листова у стаблу претраживања простора рјешења за $\Psi(l,M)$ рачунамо као број листова у под стаблу када је броду додијељена позиција са казном 0, тј. $\Psi(l-1,M)$, плус број листова у свим под стаблима са казном i, гдје може да узме вриједност од 1 до максимално M. Уколико је броду додијељена позиција са казном i, онда се број листова у под стаблу рачуна као $\Psi(l-1,M-i)$. Пошто таквих позиција има $4i$, за $i\geq 1$, онда је се укупан број таквих листова рачуна сумом $4\sum_{i=1}^{M} i\Psi(l-1,M-i)$. Повежемо ли све наведено о томе како се рачуна број листова $\Psi(l,M)$ добијамо други дио формуле (6.3.10).

Лема 6.3.1 *– Лема о* $\Psi(1,M)$

Функцију $\Psi(1,M)$ можемо израчунати као $\Psi(1,M)=2M^2+2M=1$.

Доказ.

Из дефиниције функције $\Psi(l,M)$, дате у формули 7.3.10, непосредно израчунавамо:

$$\begin{aligned}\Psi(1,M)&=\Psi(0,M)+4\sum_{i=1}^{M} i\Psi(0,M-i)=1+4\sum_{i=1}^{M} i*1=\\&=1+4\sum_{i=1}^{M} i=1+4\cdot\frac{M(M+1)}{2}=\\&=1+2M(M+1)=2M^2+2M+1.\end{aligned} \tag{6.3.11}$$

Q.E.D.

Лема 6.3.2 *– Лема о степену полинома*

Нека је $P(M)=\alpha_n M^n+\alpha_{n-1}M^{n-1}+\cdots\alpha_1 M+\alpha_0$ полином n степена по M, тј. $\deg(P)=n$. Тада је степен полинома:

$$Q(M) = \sum_{i=1}^{M} iP(M-i), \tag{6.3.12}$$

једнак $n+2$, тј. $\deg\big(Q(M)\big) = \deg\big(P(M)\big) + 2 = n+2$.

Доказ.

Развијањем израза $iP(M-i)$, у формули (6.3.12), добијамо M^n и i^{n+1}, као највеће степене промјењивих M и i. Промјењива M је слободна у изразу $iP(M-i)$, па га можемо изразити као:

$$iP(M-i) = \beta_{n+1}i^{n+1} + \beta_n i^n + \cdots \beta_1 i, \tag{6.3.13}$$

гдје су коефицијенти β_j, $j \in \{1, \dots n+1\}$ зависни од M и α_j, $j \in \{0, \dots n\}$. Полином $Q(M)$, тада можемо представити као:

$$\begin{aligned} Q(M) &= \sum_{i=1}^{M} (\beta_{n+1}i^{n+1} + \beta_n i^n + \cdots \beta_1 i) = \\ &= \beta_{n+1}\sum_{i=1}^{M} i^{n+1} + \beta_n \sum_{i=1}^{M} i^n + \cdots + \beta_1 \sum_{i=1}^{M} i. \end{aligned} \tag{6.3.14}$$

Примјетимо да коефицијент β_{n+1} не садржи M. Из дефиниције полинома $Q(M)$ непосредно слиједи:

$$Q(M) = \sum_{i=1}^{M} i(\alpha_n(M-i)^n + \alpha_{n-1}(M-i)^{n-1} + \cdots \alpha_1(M-i) + \alpha_0), \tag{6.3.15}$$

па се степен i^{n+1} јавља само у $i\alpha_n(M-i)^n$. Дизањем на n степен $(M-i)^n$ добијамо суму:

$$i\alpha_n(M-i)^n = i\alpha_n \sum_{k=0}^{n} \binom{n}{k} M^{n-k} i^k = \sum_{k=0}^{n} \binom{n}{k} \alpha_n M^{n-k} i^{k+1}. \tag{6.3.16}$$

Из суме у формули (6.3.16), јасно је да се степен i^{n+1} јавља у њеном посљедњем члану: $\binom{n}{n}\alpha_n M^{n-n} i^{n+1} = \alpha_n i^{n+1}$. Из облика посљедњег члана $\alpha_n i^{n+1}$ непосредно закључујемо да је $\beta_{n+1} = \alpha_n$, што за посљедицу има то да не зависи од M.

Како β_{n+1} не садржи M, из формуле (6.3.14) закључујемо да је $\deg\big(Q(M)\big) = \deg(\sum_{i=1}^{M} i^{n+1})$, гдје рачунамо степен полином по промјењивој. Користећи

познату *Фулхаберову формулу* за суму $n+1$ степена првих M, природних бројева добијамо формулу:

$$\sum_{i=1}^{M} i^{n+1} = \frac{1}{n+2}\sum_{j=0}^{n+1}(-1)^j \binom{n+2}{j} B_j M^{n+2-j}, \tag{6.3.17}$$

где су B_j *први Бернулијеви* бројеви, са вриједношћу $B_1 = -1/2$. Детаљи о *Фулхаберовој формули* могу се наћи у (Conway & Guy, 1996). Из формуле 6.3.17 непосредно изводимо:

$$\deg\big(Q(M)\big) = \deg\left(\sum_{i=1}^{M} i^{n+1}\right) = \deg(M^{n+2}) = n+2. \quad \textbf{Q.E.D.} \tag{6.3.18}$$

Лема 6.3.3 *– Лема о степену* $\Psi(l, M)$

Степен $\Psi(l, M)$ по промјенљивој M, је $2l$, тј. $\deg\big(\Psi(l, M)\big) = 2l$, за $M \geq 0$.

Доказ.

Доказ изводимо примјеном принципа математичке индукције по промјенљивој l.

(1) За $l = 1$, на основу леме 7.3.1 имамо да је $\Psi(1, M) = 2M^2 + 2M = 1$, одакле непосредно закључујемо да је $\deg\big(\Psi(1, M)\big) = 2$.

(2) *Индуктивна хипотеза:* претпоставимо да важи $\deg\big(\Psi(l, M)\big) = 2l$, за $M \geq 0$.

(3) *Индуктивни корак:* докажимо да важи $\deg\big(\Psi(l+1, M)\big) = 2(l+1)$, за $M \geq 0$.

На основу дефиниције функције Ψ, дате у формули (6.3.10), важи:

$$\Psi(l+1, M) = \Psi(l, M) + 4\sum_{i=1}^{M} i\Psi(l, M-i). \tag{6.3.19}$$

На основу индуктивне хипотезе закључујемо да:

$$\deg\big(\Psi(l, M)\big) = 2l \ \wedge \ (\forall i \in \{1, \dots, M\})\ \deg\big(\Psi(l, M-i)\big) = 2l. \tag{6.3.20}$$

На основу леме 6.3.2 имамо да је:

$$\deg\left(\sum_{i=1}^{M} i\Psi(l, M-i)\right) = 2l + 2. \tag{6.3.21}$$

Коначно, закључујемо:

$$\begin{aligned}\deg\big(\Psi(l+1, M)\big) = \\ = \max\left(\deg\big(\Psi(l, M)\big), \deg\left(\sum_{i=1}^{M} i\Psi(, M-i)\right)\right) \\ = \max(2l, 2l+2) = 2l+2 = 2(l+1).\ \textbf{Q.E.D.}\end{aligned} \tag{6.3.22}$$

Теорема 6.3.1 *– Процјена комплексности SEDA за BAP*
Комплексност SEDA-а за рјешавање BAP-а је $O(M^{2l})$.

Доказ.
Доказ је непосредна посљедица леме 6.3.3, у којој је доказано да је $\Psi(l, M)$ полином по M, степена $2l$, тј. комплексност SEDA-а за BAP је $O(M^{2l})$.
Q.E.D.

Комплексност $O(M^{2l})$ SEDA постиже се на најгори сценарио улазних података. У изведеној процјени занемарили смо казну кашњења испловљавања брода. Увртавањем казне кашњења испловљавања брода, комплексност би била снижена, али не значајно. Због тога можемо сматрати да је општа комплексност SEDA-а за DBAP $O(M^{2l})$. У овој процјени, такође су занемарене границе хоризонта планирања и броја везова. Њихове димензије су коначне, што такође снижава комплексност израчунавања SEDA-а у конкретним примјерима. Горња граница се достиже у примјерима са великим бројем бродова, о чему ће бити више ријечи када будемо комнетарисали експерименталне резултате у подпоглављу 7.4.

Комплексност зависи од процјене оптималног рјешења M. Квалитет процјене, или првог допустивог рјешења, утиче на ефикасност SEDA-а. У процјени комплексности подразумијевали смо да је вриједност за M константна. То није случај у конкретним примјерима, јер у њима улогу процјене M, узима вриједност тренутно најбољег рјешења, током рада SEDA-а. Током рада, SEDA најчешће веома брзо дође до оптималног рјешења, а највећи дио времена се утроши на то да се *„докаже“* да не постоји боље рјешење.

Због тога, квалитет процјене оптималног рјешења M је важан, али није од пресудиног значаја за ефикасност SEDA-а.

***Теорема 6.3.2** – Процјена комплексности SEDA+ERH за BAP*

Ако са ω означимо редослијед бродова до којег је дошао ERH, са o оптимално рјешење проблема и ако k_ω дефинишемо као:

$$k_\omega = \max\{\, i \in L \mid f_{\omega(i)}(o_i) > 0\}, \tag{6.3.23}$$

тада комплексност SEDA+ERH можемо изразити као $O(M^{2k_\omega})$.

Доказ.

ERH поставља на почетак у пермутацији бродова ω, оне бродове за које се плаћа казна. То су они бродови, који доприносе вриједностима различитим од нуле у укупној суми функције критеријума. Због тога што су на почетку, они добијају раније позиције на којима се плаћа казна, и због тога смањују простор претраживања за бродове који слиједе након њих, остављајући им позиције са мањим казнама. У тренутку када је вриједност оптимума досегнута, код брода k_ω, тада једине допустиве позиције за преосталих $l - k_\omega \geq 0$ бродова, су оне са казном 0. Због тога, конструише са само једна грана максималне дужине, чиме се окончава претрага за оптималним рјешењем. То значи да је максимални број бродова које је досегао SEDA за поредак бродова ω управо k_ω. На основу теореме 6.3.1, непосредно закључујемо да је комплексност SEDA+ERH можемо изразити као $O(M^{2k_\omega})$. **Q.E.D.**

Уводимо количник $D_\omega = l/k_\omega$, којег називамо *коефицијентом успорења*. Користећи њега комплексност SEDA+ERH можемо изразити као $O\left(M^{2l/D_\omega}\right)$. Најчешће, у конкретним примјерима, D_ω узима вриједности у опсегу од 2 до 5, што објашњава далеко бољу ефикасност SEDA+ERH у односу на SEDA.

***Теорема 6.3.3** – Процјена комплексности SEDA+ERH+D&C за BAP*

Ако означимо подпроблеме који су рјешавани, сходно опису алгоритма 4.3.1, током примјене D&C стратегије са $\omega_1, \dots, \omega_N \subseteq L$, за неко $N \geq 1$. Изаберимо међу њима $\sigma \in \{\omega_1, \dots, \omega_N\}$, тако да је он највеће кардиналности, тј. да важи:

$$(\forall i \in \{1, \dots, N\})\ |\omega_i| \leq |\sigma|. \tag{6.3.24}$$

Тада јкомплексност SEDA+ERH+D&C можемо процијенити као $O\left(NM^{2|\sigma|}\right)$.

Доказ.

Непосредно слиједи из теореме 6.3.2 и чињенице да покрећемо највише N, пута SEDA+ERH, сваки пут са скупом бродова чија је кардиналност мања или једнака од $|\sigma|$. **Q.E.D.**

На крају разматрања комплексности примјетимо да су комплексности сва три алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C експоненцијална: $O(M^{2l})$, $O(M^{2k_\omega})$, и $O\left(NM^{2|\sigma|}\right)$. У све три оцјене зависне су од константе M, којом процјењујемо вриједност оптимума. Уколико је процјена добра, и вриједности оптимума мала, то за посљедицу има могућност проналаска оптималног рјешење, без обзира, колико је велик број бродова. Сматрамо да је то веома важно својство сва три наведена алгоритма.

6.4. ЕКСПЕРИМЕНТАЛНИ РЕЗУЛТАТИ

У овом подпоглављу даћемо прво опис тест примјера, затим упоређење и анализу предложених алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за рјешавање DBAP-а и HBAP-а, и на крају извршити упоређење са CPLEX-ом и *state-of-art* приступима рјешавања DBAP-а и HBAP-а.

6.4.1. Опис тест примјера

Експерименти су спроведени на три класе проблема, који су слични по димензијама онима који су уведени у (Giallombardo, Moccia, Salani, & Vacca, 2010). Разматраћемо тест примјере са 5, 8 и 13 везова, а хоризонт планирања биће 1 или 2 седмице. Хоризонт планирања је подијељен а јединице у трајању од 3 сата, па тако хоризонт од 1 седмице има 56 временских јединица, а хоризонт од 2 седмице има 112 временске јединице. Број бродова ће ићи од 5 бродова, па све до 120 бродова у неким примјерим, у корацима од 5 бродова. Преглед димензија тест примјера дат је у табели 6.4.1.

Табела 6.4.1 *– Преглед димензија тест примјера за DABP и HBAP*

Класа	***Број везова*** m	***Хоризонт планирања*** T	***Број бодова*** l
I	5	1 седмица	од 5 до 45
II	8	2 седмице	од 5 до 120
III	13	2 седмице	од 5 до 80

Информације, које се односе на типове бродова, који се јављају у тест примјерима дати су у табели 6.4.2. Спецификација је преузета из (Meisel, Seaside Operations Planning in Container Terminals, 2009) и (Bierwirth & Meisel, 2010). Постоје три типа бродова: *мали, средњи* и *велики*. За сваки тип дата је њихов проценат у тест примјерима, трајање веза док се обаве операције искрцаја и утовара брода, казне за прекорачења (сходоно потпоглављу 6.1.4) у јединицама од 1000$ и број везова који заузимају.

***Табела 6.4.2** – Преглед димензија тест примјера за DABP и HBAP*

Тип брода	*Проценат у примјерима*	*Дужина обраде*	*C1*	*C2*	*C3*	*C4*	*Број везова DBAP*	*Број везова HBAP*
Мали	60%	1—3	2	3	3	9	1	1
Средњи	30%	4—5	3	6	6	18	1	2
Велики	10%	6—8	4	9	9	27	1	3

Дистрибуција најјефтинијих везова је хомогена у оквиру тест полулације бродова. За сваки облик теста, случајно је генерисано по 500 примјера. Регистровали смо проценат примјера, у оквиру теста, који су ријешени за мање од пола сата. Примјерима, који су ријешени за мање од пола сата, биљежили смо и најкраће, просјечно и најдуже вријеме рјешавања. Они други примјери, који нијесу ријешени за пола сата, су прекидани са израдом и њихово израчунавање није ушло у евиденцију најкраћих, средњих и најдужих времена. Сва времена у таблицама изражена су у *секундама*.

6.4.2. Међусобно поређење предложених алгоритама

Алгоритами SEDA, SEDA+ERH и SEDA+ERH+D&C програмски су имплементирани у програмском језику *Wolfram Language* и у програмском језику *C*. Овдје ће бити приказани резултати програмске имплементације у *C* програмском језику. C компајлер који је коришћен је: *Microsoft C/C++ Optimizing Compiler version 18.00.31101 for x86*.

Тестови су извршавани на рачунару са *Intel Core i7-4500U @* 1.80GHz—2.40GHz процесором, 8 GB RAM меморије којим је управљао *Microsoft Windows 8.1* 64-bit оперативни систем.

У свим табелама са упоредним временима, цртице у рубрикама за времена означавају да је тестови нијесу извођени, због спорости алгоритам. Прво су изложени експериментални резултати за Класу I и II, у случају DBAP-а.

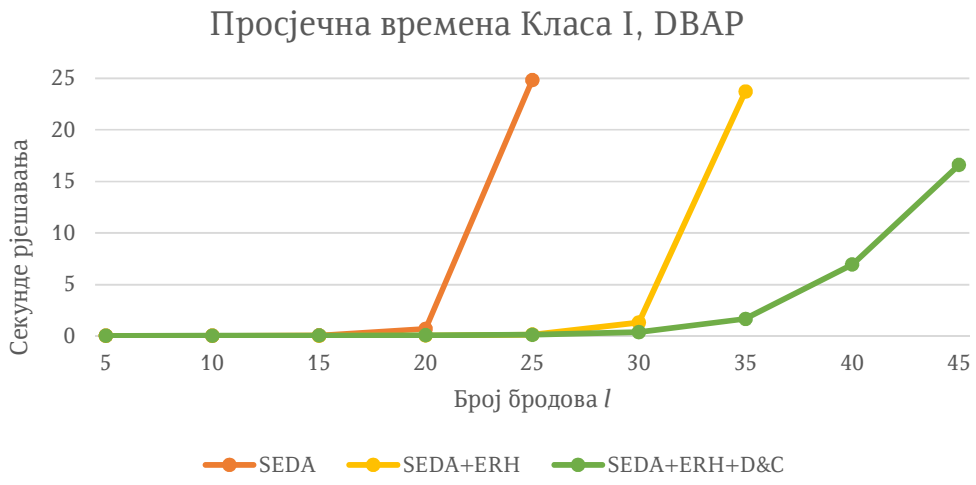
Табела 6.4.3 *– Упоредна времена извршавања тестова алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за Класи I, врста проблема DBAP*

	Класа I | 5×56 | 500 примјера | DBAP											
	SEDA				*SEDA+ERH*				*SEDA+ERH+D&C*			
l	t ≤ ½h %	min	**avg**	max	t ≤ ½h %	min	**avg**	max	t ≤ ½h %	min	**avg**	max
5	*100.0*	0.00	**0.02**	0.05	*100.0*	0.00	**0.02**	0.07	*100.0*	0.00	**0.02**	0.05
10	*100.0*	0.02	**0.03**	0.08	*100.0*	0.02	**0.03**	0.05	*100.0*	0.01	**0.04**	0.08
15	*100.0*	0.02	**0.05**	1.53	*100.0*	0.02	**0.04**	0.16	*100.0*	0.02	**0.06**	0.19
20	*99.8*	0.02	**0.71**	64.00	*100.0*	0.02	**0.06**	0.22	*100.0*	0.02	**0.08**	0.14
25	*96.8*	0.02	**24.83**	1158.46	*100.0*	0.03	**0.15**	7.41	*100.0*	0.03	**0.13**	1.46
30	–	–	–	–	*100.0*	0.04	**1.33**	80.45	*100.0*	0.08	**0.39**	11.26
35	–	–	–	–	*96.4*	0.06	**23.73**	1382.63	*100.0*	0.09	**1.68**	210.37
40	–	–	–	–	–	–	–	–	*100.0*	0.11	**6.94**	1325.05
45	–	–	–	–	–	–	–	–	*99.4*	0.16	**16.60**	1077.15

У табели 6.4.3 дата су упоредна времена извршавања алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за рјешавање DBAP-а. Тест примјеру су Класе I, што значи да лука има 5 везова и да се план прави за 1 седмицу, тј. 56 временских јединица од 3 сата. SEDA је успио да ријеши свих 500 примјера у року од пола сата за примјере са од 5 до 15 бродова, SEDA+ERH је то пошло за руком за примјере од 5 до 25 борова, значи за 10 бродова више у односу на SEDA-у. Коначно SEDA+ERH+D&C је ријешио свих 500 примјера за мање од пола сата за примјере од 5 до 40 бродова, дакле 20 бродова више од SEDA-а и 10 бродова више од SEDA+ERH.

Сва три алгоритма се понашају приближно једнако за примјере од 5 до 10 бродова. За примјере са 15 и више бродова SEDA значајно заостаје за SEDA+ERH и SEDA+ERH+D&C. У примјерима са 20 бродова SEDA+ERH и SEDA+ERH+D&C су приближно **165** бута бржи од SEDA-а. За 30 и више бродова предност SEDA+ERH+D&C постаје очигледна.

У примјерима са 30 бродова SEDA+ERH+D&C је 3.41 пут бржи од SEDA+ERH, док је у примјерима са 35 бродова **14.1** пута бржи. Графикон средњих брзина израчунавања приказан је на илустрацији 6.4.1. Примјетимо и то да је просјечно вријеме рјешавања SEDA+ERH+D&C за примјере са 45 бродова 16.60 *sec*, мање од просјечног времена рјешавања SEDA+ERH за 35 бродова и просјечног времена рјешавања SEDA-а за 25 бродова.



Илустрација 6.4.1 *– Просјечне брзине рјешавање проблема Класе I, DBAP-а.* На хоризонталној оси се налази број бодова у примјерима Класе I, док на вертикалној оси налазе просјечна времена рјешавања проблема.

У табели 6.4.4 дата су упоредна времена извршавања алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за рјешавање DBAP-а. Тест примјеру су Класе II, што значи да лука има 8 везова и да се план прави за 2 седмице, тј. 112 временских јединица од 3 сата. SEDA је успио да ријеши свих 500 примјера у року од пола сата за примјере са од 5 до 30 бродова, SEDA+ERH је то пошло за руком за примјере од 5 до 60 борова, значи за 30 бродова више у односу на SEDA. Коначно SEDA+ERH+D&C је ријешио свих 500 примјера за мање од пола сата за примјере од 5 до 100 бродова, дакле 70 бродова више од SEDA и 40 бродова више од SEDA+ERH.

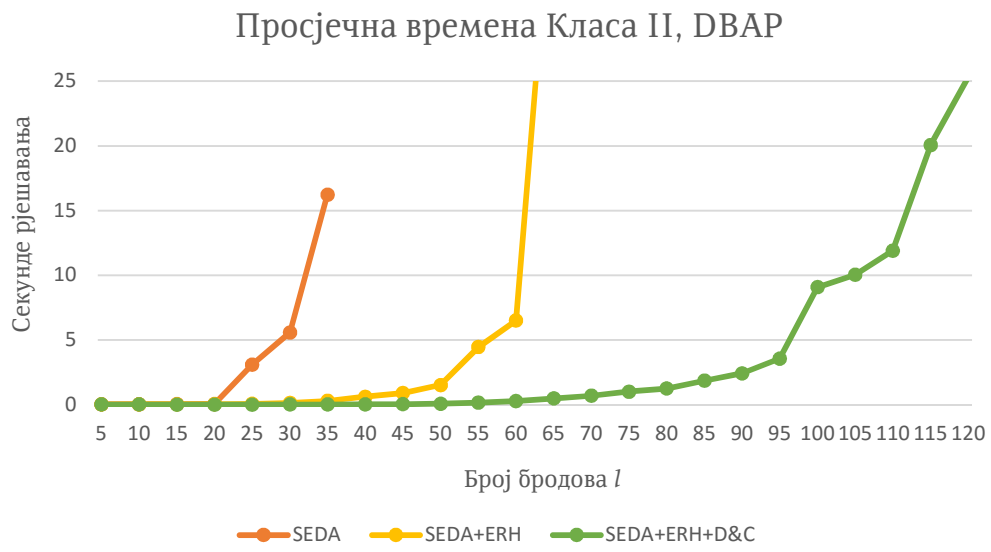
Сва три алгоритма се понашају приближно једнако за примјере од 5 до 10 бродова. За примјере са 25 и више бродова SEDA значајно заостаје за SEDA+ERH и SEDA+ERH+D&C. У примјерима са 25 бродова SEDA+ERH је приближно **44** пута бржи од SEDA, док је SEDA+ERH+D&C за исте случаје приближно **310** бута бржи од SEDA. За 35 и више бродова предност SEDA+ERH+D&C постаје очигледна. У примјерима са 40 бродова SEDA+ERH+D&C је **20** пута бржи од SEDA+ERH, док је у примјерима са 60 бродова **23** пута бржи.

Табела 6.4.4 *– Упоредна времена извршавања тестова алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за Класи II, врста проблема DBAP*

	Класа II | 8×112 | 500 примјера | DBAP											
	SEDA				*SEDA+ERH*				*SEDA+ERH+D&C*			
l	t ≤ ½h %	min	**avg**	max	t ≤ ½h %	min	**Avg**	max	t ≤ ½h %	min	**avg**	max
5	*100.0*	0.00	**0.03**	0.10	*100.0*	0.01	**0.02**	0.04	*100.0*	0.00	**0.02**	0.05
10	*100.0*	0.02	**0.03**	0.11	*100.0*	0.02	**0.03**	0.05	*100.0*	0.02	**0.02**	0.06
15	*100.0*	0.02	**0.04**	0.18	*100.0*	0.02	**0.03**	0.10	*100.0*	0.00	**0.01**	0.08
20	*100.0*	0.03	**0.06**	4.12	*100.0*	0.02	**0.05**	0.49	*100.0*	0.00	**0.01**	0.03
25	*100.0*	0.03	**3.10**	1278.34	*100.0*	0.04	**0.07**	0.78	*100.0*	0.00	**0.01**	0.02
30	*100.0*	0.04	**5.58**	1127.44	*100.0*	0.04	**0.14**	0.85	*100.0*	0.00	**0.02**	0.11
35	*98.0*	0.02	**16.23**	998.81	*100.0*	0.04	**0.30**	1.47	*100.0*	0.00	**0.02**	0.16
40	–	–	–	–	*100.0*	0.05	**0.61**	1.88	*100.0*	0.01	**0.03**	0.38
45	–	–	–	–	*100.0*	0.05	**0.91**	2.28	*100.0*	0.02	**0.04**	0.36
50	–	–	–	–	*100.0*	0.05	**1.52**	22.10	*100.0*	0.02	**0.08**	0.66
55	–	–	–	–	*100.0*	0.07	**4.46**	385.48	*100.0*	0.02	**0.16**	0.95
60	–	–	–	–	*100.0*	0.24	**6.51**	780.01	*100.0*	0.03	**0.28**	1.23
65	–	–	–	–	*99.4*	0.41	**41.82**	1753.07	*100.0*	0.03	**0.49**	1.38
70	–	–	–	–	–	–	–	–	*100.0*	0.03	**0.70**	3.88
75	–	–	–	–	–	–	–	–	*100.0*	0.05	**1.01**	22.74
80	–	–	–	–	–	–	–	–	*100.0*	0.05	**1.25**	12.09
85	–	–	–	–	–	–	–	–	*100.0*	0.06	**1.86**	26.05
90	–	–	–	–	–	–	–	–	*100.0*	0.58	**2.43**	19.27
95	–	–	–	–	–	–	–	–	*100.0*	1.23	**3.56**	38.16
100	–	–	–	–	–	–	–	–	*100.0*	1.81	**9.08**	984.72
105	–	–	–	–	–	–	–	–	*99.6*	1.91	**10.04**	1153.83
110	–	–	–	–	–	–	–	–	*99.2*	2.13	**11.91**	1199.67
115	–	–	–	–	–	–	–	–	*99.2*	2.41	**20.05**	1126.23
120	–	–	–	–	–	–	–	–	*97.4*	2.61	**25.30**	1566.79

Наставимо даље са експерименталним резултатима за Класу II и III, у случају HBAP. У табели 6.4.5 дата су упоредна времена извршавања алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за рјешавање HBAP. Тест примјеру су Класе II, што значи да лука има 8 везова и да се план прави за 2 седмице, тј. 112 временских јединица од 3 сата. SEDA је успио да ријеши свих 500 примјера у року од пола сата за примјере са од 5 до 15 бродова, SEDA+ERH је то пошло за руком за примјере од 5 до 20 борова, значи за 5 бродова више у односу на SEDA. Коначно SEDA+ERH+D&C је ријешио свих

500 примјера за мање од пола сата за примјере од 5 до 50 бродова, дакле 35 бродова више од SEDA и 30 бродова више од SEDA+ERH.



Илустрација 6.4.2 *– Просјечне брзине рјешавање проблема Класе II, DBAP-а.* На хоризонталној оси се налази број бодова у примјерима Класе II, док на вертикалној оси налазе просјечна времена рјешавања проблема.

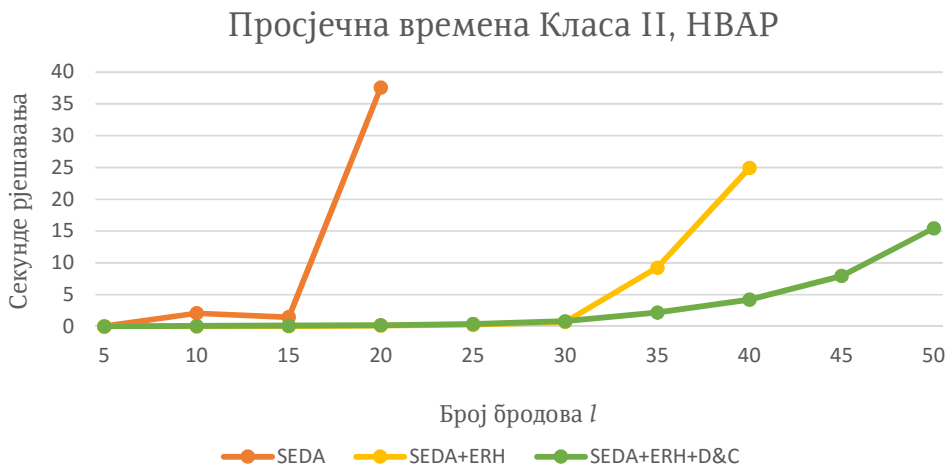
Наставимо даље са експерименталним резултатима за Класу II и III, у случају HBAP. У табели 6.4.5 дата су упоредна времена извршавања алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за рјешавање HBAP. Тест примјеру су Класе II, што значи да лука има 8 везова и да се план прави за 2 седмице, тј. 112 временских јединица од 3 сата. SEDA је успио да ријеши свих 500 примјера у року од пола сата за примјере са од 5 до 15 бродова, SEDA+ERH је то пошло за руком за примјере од 5 до 20 борова, значи за 5 бродова више у односу на SEDA. Коначно SEDA+ERH+D&C је ријешио свих 500 примјера за мање од пола сата за примјере од 5 до 50 бродова, дакле 35 бродова више од SEDA и 30 бродова више од SEDA+ERH.

Као и у случају DBAP, јасна је надмоћ SEDA+ERH+D&C у односу на SEDA и SEDA+ERH. Она се у овом примјеру посебно огледа у томе што је SEDA+ERH+D&C алгоритам у стању да комплетно ријеши далеко већи број примјера са више бродова: 50 у односу на 15 и 20 колико постижу SEDA и SEDA+ERH. За мањи број бродова та предност није уочљива, чак су просјечна времена SEDA+ERH+D&C већа од просјечних времена SEDA и SEDA+ERH.

***Табела 6.4.5** – Упоредна времена извршавања тестова алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за Класи II, врста проблема HBAP*

	Класа II | 8×112 | 500 примјера | HBAP											
	SEDA				*SEDA+ERH*				*SEDA+ERH+D&C*			
l	t ≤ ½h %	min	**avg**	max	t ≤ ½h %	min	**avg**	max	t ≤ ½h %	min	**avg**	max
5	*100.0*	0.02	**0.02**	0.04	*100.0*	0.02	**0.02**	0.03	*100.0*	0.02	**0.04**	0.36
10	*100.0*	0.02	**2.09**	743.65	*100.0*	0.02	**0.03**	0.16	*100.0*	0.04	**0.07**	1.63
15	*100.0*	0.02	**1.47**	227.43	*100.0*	0.02	**0.06**	0.47	*100.0*	0.01	**0.13**	1.97
20	*97.2*	0.02	**37.59**	1761.46	*100.0*	0.02	**0.14**	0.96	*100.0*	0.05	**0.20**	4.28
25	–	–	–	–	*99.8*	0.02	**0.26**	2.18	*100.0*	0.07	**0.39**	10.16
30	–	–	–	–	*99.8*	0.03	**0.71**	32.57	*100.0*	0.09	**0.83**	12.69
35	–	–	–	–	*99.6*	0.04	**9.26**	1312.61	*100.0*	0.08	**2.21**	19.64
40	–	–	–	–	*97.0*	0.05	**24.94**	1462.37	*100.0*	0.10	**4.21**	42.25
45	–	–	–	–	–	–	–	–	*100.0*	0.04	**7.96**	35.60
50	–	–	–	–	–	–	–	–	*100.0*	0.12	**15.44**	1103.45
55	–	–	–	–	–	–	–	–	*99.60*	0.17	**11.87**	1074.94

Међутим, већ од примјера са 35 бродова, SEDA+ERH+D&C има значајно краће просјечно и максимално забиљежено вријеме рјешавања. Смањење просјечног и максимално забиљеженог времена рјешавања SEDA+ERH+D&C, у случају са 55 бродова, је посљедица не узимања у просјек 2 примјера (0.4%) чије је рјешавање трајало дуже од пола сата.



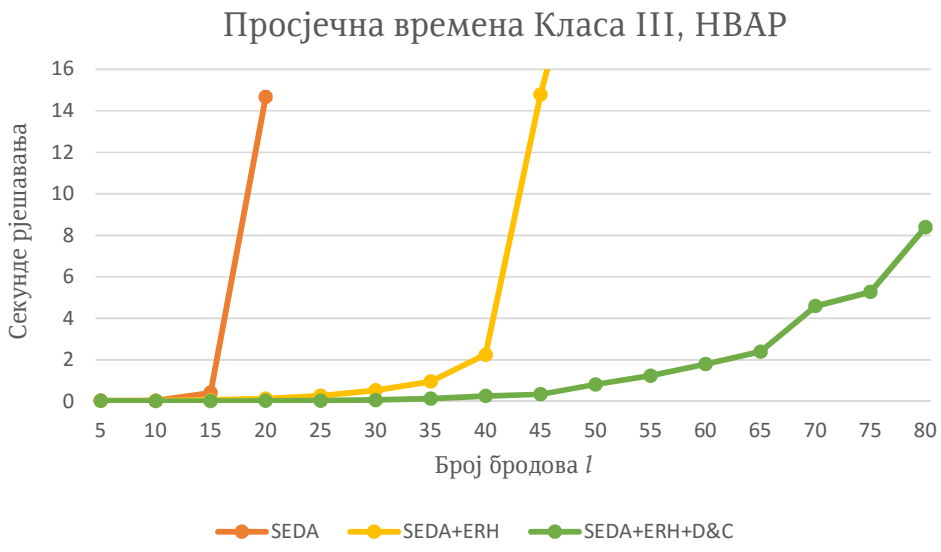
***Илустрација 6.4.3** – Просјечне брзине рјешавање проблема Класе II, HBAP.* На хоризонталној оси се налази број бодова у примјерима Класе II, док на вертикалној оси налазе просјечна времена рјешавања проблема.

***Табела 6.4.6** – Упоредна времена извршавања тестова алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за Класи III, врста проблема HBAP*

	Класа III | 13×112 | 500 примјера | HBAP											
	SEDA				*SEDA+ERH*				*SEDA+ERH+D&C*			
l	t ≤ ½h %	min	**avg**	max	t ≤ ½h %	min	**avg**	max	t ≤ ½h %	min	**avg**	max
5	*100.0*	0.01	**0.02**	0.07	*100.0*	0.01	**0.02**	0.08	*100.0*	0.01	**0.03**	0.08
10	*100.0*	0.02	**0.03**	1.57	*100.0*	0.02	**0.03**	0.13	*100.0*	0.01	**0.01**	0.10
15	*100.0*	0.02	**0.41**	131.87	*100.0*	0.02	**0.06**	0.63	*100.0*	0.01	**0.01**	0.04
20	*98.6*	0.02	**14.66**	1756.80	*100.0*	0.02	**0.13**	1.19	*100.0*	0.01	**0.02**	0.11
25	–	–	–	–	*100.0*	0.04	**0.27**	1.96	*100.0*	0.02	**0.03**	0.52
30	–	–	–	–	*100.0*	0.04	**0.53**	2.79	*100.0*	0.01	**0.06**	0.65
35	–	–	–	–	*100.0*	0.05	**0.95**	4.13	*100.0*	0.02	**0.13**	9.95
40	–	–	–	–	*100.0*	0.11	**2.25**	66.05	*100.0*	0.25	**0.25**	15.47
45	–	–	–	–	*99.8*	0.12	**14.79**	1434.77	*100.0*	0.38	**0.34**	2.52
50	–	–	–	–	*98.2*	0.19	**24.89**	1338.50	*100.0*	0.03	**0.81**	41.39
55	–	–	–	–	–	–	–	–	*100.0*	0.03	**1.23**	24.33
60	–	–	–	–	–	–	–	–	*100.0*	0.05	**1.79**	21.31
65	–	–	–	–	–	–	–	–	*100.0*	0.05	**2.39**	50.54
70	–	–	–	–	–	–	–	–	*100.0*	0.07	**4.59**	342.51
75	–	–	–	–	–	–	–	–	*100.0*	0.98	**5.27**	296.84
80	–	–	–	–	–	–	–	–	*99.4*	1.53	**8.39**	494.39

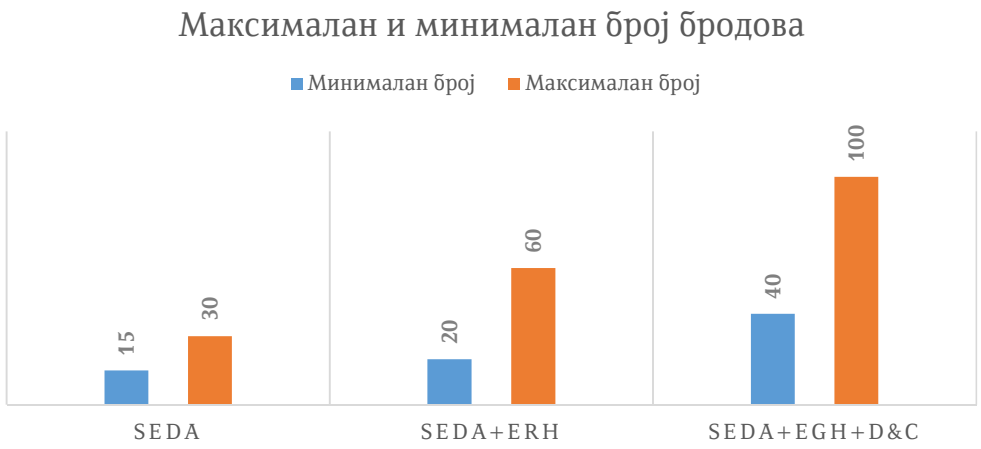
У табели 6.4.6 дата су упоредна времена извршавања алгоритама SEDA, SEDA+ERH и SEDA+ERH+D&C за рјешавање HBAP-а, проблема Класе III. Лука има 8 везова, а хоризонт планирања је 2 седмице, тј. 112 временских јединица од 3 сата. SEDA је успио да ријеши свих 500 примјера у року од пола сата за примјере са од 5 до 15 бродова, SEDA+ERH је то пошло за руком за примјере од 5 до 40 борова, значи за 25 бродова више у односу на SEDA. Коначно SEDA+ERH+D&C је ријешио свих 500 примјера за мање од пола сата за примјере од 5 до 75 бродова, дакле 60 бродова више од SEDA и 35 бродова више од SEDA+ERH.

Као и претходним примјерима, јасна је надмоћ SEDA+ERH+D&C у односу на SEDA и SEDA+ERH. Она се и у овом примјеру посебно огледа у томе што је SEDA+ERH+D&C алгоритам у стању да комплетно ријеши далеко већи број примјера са више бродова: 75 у односу на 15 и 40 колико постижу SEDA и SEDA+ERH. Поново, за мањи број бродова та предност није толико уочљива. Међутим, већ од примјера са 40 бродова, SEDA+ERH+D&C има значајно краће просјечно и максимално забиљежено вријеме рјешавања.



Илустрација 6.4.4 *– Просјечне брзине рјешавање проблема Класе III, HBAP-а.* На хоризонталној оси се налази број бодова у примјерима Класе III, док на вертикалној оси налазе просјечна времена рјешавања проблема.

Сумирајмо претходне примјере у графикону у којем ћемо приказати минималан и максималан број бродова, за које су алгоритми SEDA, SEDA+ERH и SEDA+ERH+D&C успјели да ријеше свих 500 примјера у оквиру временског ограничења од пола сата за рјешавање појединог примјера, датим у илустрацији 6.4.5.



Илустрација 6.4.5 *– Максималан и минималан и максималан број бродова, за које су алгоритми SEDA, SEDA+ERH и SEDA+ERH+D&C успјели да ријеше свих 500 примјера у оквиру временског ограничења од пола сата за рјешавање појединог примјера.*

Основни SEDA алгоритам, очекивано, остварује најслабије резултате у свим експериментима успио је да ријеши свих 500 примјера за број бродова који

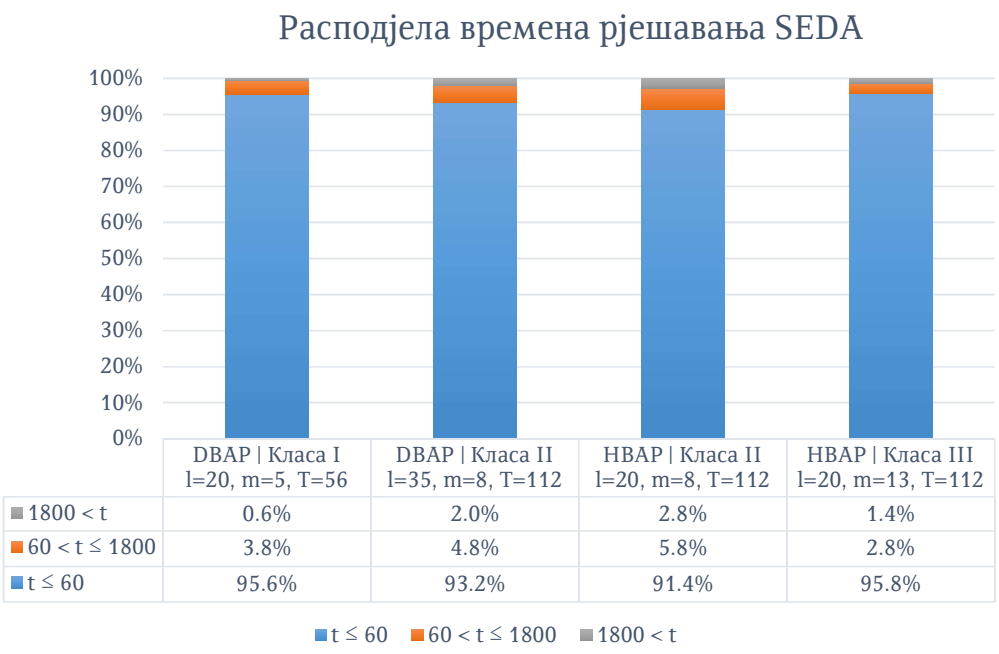
је био у опсегу од 15 до 20 бродова. SEDA+ERH алгоритам, је остварио боље резултате, рјешавајући свих 500 примјера, за број бродова који је био у опсегу од 20 до 60 бродова. Најбољи резултат је постигао SEDA+ERH+D&C алгоритам, који је успио да ријеши свих 500 примјера, за број бродова који је био у опсегу од 40 до 100 бродова.

Најтежи примјер је сва три алгоритма је DBAP, Класа I, тј. када рјешавамо дискретан случај BAP-а у луци са 5 везова и хоризонтом планирања од 1 седмице, односно 56 временских јединица. Овај проблем је најтежи, зато што већ код малог броја бродова имамо велики број њих, који су у конфликту. У осталим примјерима, тек са већим бројем бродова, долазимо у ситуацију да је велики број бродова у конфликту.

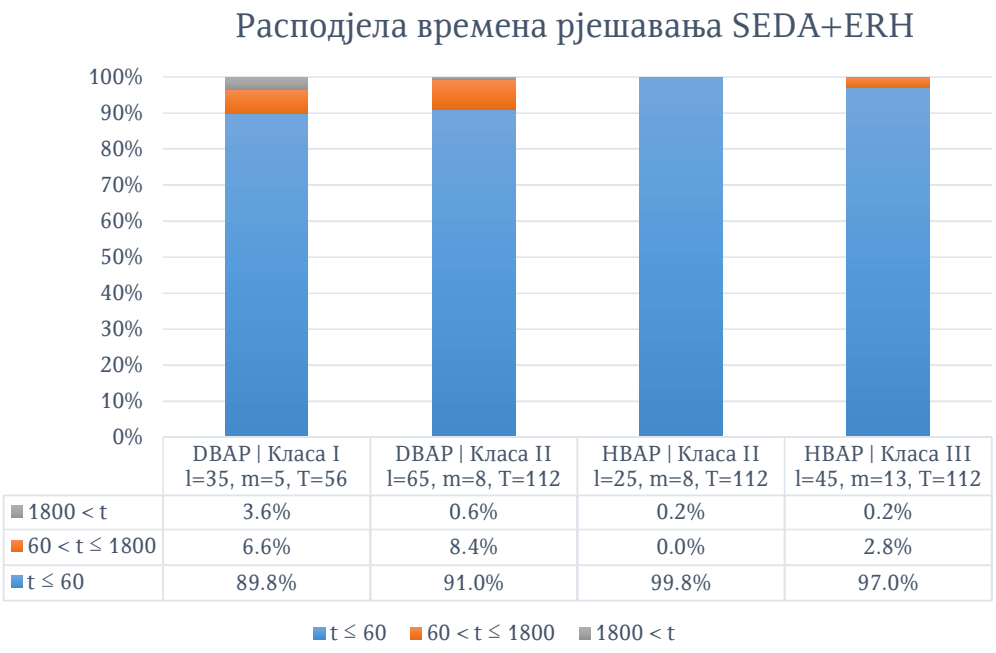
Размотримо даље распоредјелу времена рјешавања за алгоритме SEDA, SEDA+ERH и SEDA+ERH+D&C. За сва три алгоритма посматраћемо, у сва четири примјера, први најмањи број бродова за које је алгоритам није успио да ријеши свих 500 примјера у оквиру ограничења од пола сата. Регистроваћемо проценат примјера који су ријешени за мање од минут времена, они који су ријешени у интервалу од минут до пола сата и коначно проценат оних примјера, који нијесу ријешени у периоду од пола сата.

На илустрацији 6.4.6 приказана је расподјела за SEDA. Из ње видимо да је проценат ријешених за минут најмањи у случају HBAP-а, Класа II, и износи 91.4%, што је висок проценат. SEDA+ERH проценат ријешених за минут најмањи је у случају DBAP-а, Класа I, и износи 89.9%, што је приказано на илустрацији 6.4.7. Коначно, SEDA+ERH+D&C најмање је ријешио примјера у случају DBAP, Класа I, укупно 95.6%, за минут времена, што је приказано на илустрацији 6.4.8.

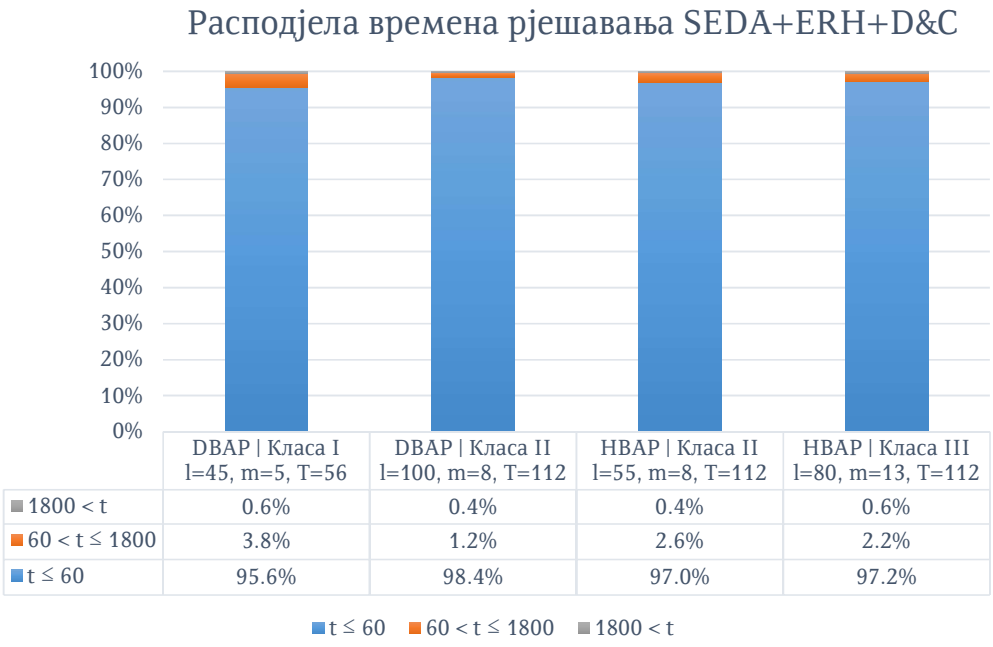
Сматрамо да је способност сва три алгоритма, да ако могу да ријеше проблем, раде то у великом проценту за најдуже минут времена, веома битно својство *методе седиментације*. Методи за рјешавање BAP базирани на њему, ако могу, могу веома брзо да дођу до рјешења.



Илустрација 6.4.6 *– Расподјела времена рјешавања SEDA*. Приказана је расподјела на класама са минималним бројем бродова, које SEDA није успјела да риши за свих 500 примјера имајући ограничење од 1800 секунди за рјешење сваког понаособ примјера.



Илустрација 6.4.7 *– Расподјела времена рјешавања SEDA+ERH*. Приказана је расподјела на класама са минималним бројем бродова, које SEDA+ERH није успјела да риши за свих 500 примјера имајући ограничење од 1800 секунди за рјешење сваког понаособ примјера.



Илустрација 6.4.8 *– Расподјела времена рјешавања SEDA+ERH+D&C.* Приказана је расподјела на класама са минималним бројем бродова, које SEDA+ERH+D&C није успјела да риши за свих 500 примјера имајући ограничење од 1800 секунди за рјешење сваког понаособ примјера.

Ова њихова способност може бити од значаја за будући развој алгоритама за рјешавање BAP. На примјер: солвер базиран на *методи седиментације* се пусти да временски ограничено тражи рјешење, уколико не дође до рјешења у предвиђеном року, пробати ријешити проблем неким другим методом.

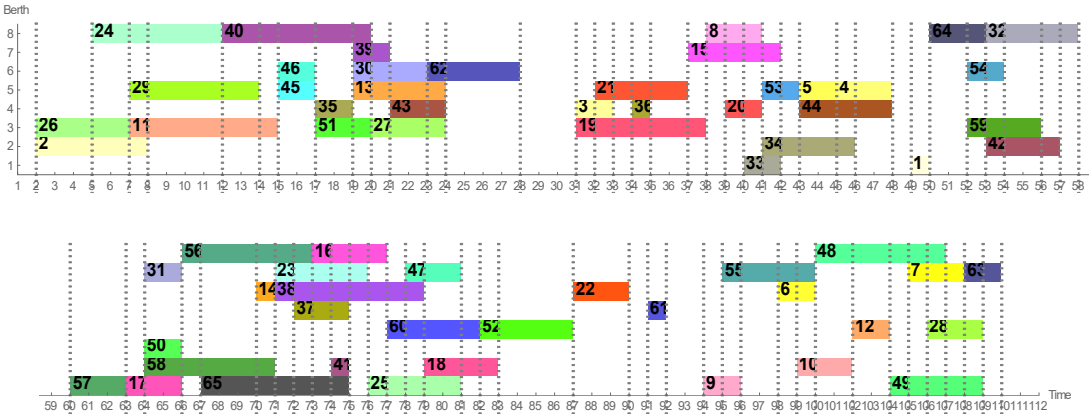
6.4.3. Анализа неколико тешких инстанци

У овом подпоглављу анализираћемо неколико тешких инстанци BAP-а. Под тешким подразумијева се да су то инстанце, чије је рјешавање узело више од пола сата, што је било ограничење за претходне примјере BAP. Анализа је урађена само за алгоритме SEDA+ERH и SEDA+ERH+D&C.

Прво дискутујемо DBAP инстанцу бр. 76, Класе II, за 8 везова и 112 временских јединица. Њено оптимално рјешење приказано је на илустрацији 6.4.9. Вриједност функције критеријума за оптимално рјешење је 122 (122 000$).

Опишимо укратко како је ову инстанцу рјешавао SEDA+ERH. Приликом првог покушаја да ријеши ову инстанцу, ERH хеуристика је започела процјену са горњим ограничењем од 206 за вриједност оптимума. По завршетку рада ERH хеуристика је процјенила да оптимално рјешење може да има

вриједност 122 или мању од ње. До те процјене је дошла након 2.60 секунде рада. Потом је SEDA покушао да поправи резултат ERH, али није успио да претражи простор допустивих рјешења са ограничењем 122 за мање од пола сата. Исти експеримент је поновљен у другом покушају, без временског ограничења и SEDA било потребно 2698.728 секунда да дође до одговора тј. да претражи простор допустивих рјешења са ограничењем 122.



Илустрација 6.4.9 *– Оптимално рјешење проблема бр. 76, са 65 бродова, 8 везова, и 112 временске јединице.* Временска зона је подијељена у два дијела. Бродови су представљени као правоугаоници у вез×вријеме координатном систему (матрици). Сваки брод има свој идентификатор у лијевом углу правоугаоника који га представља.

Оно што чини инстанцу бр. 76 тешком за рјешавање је 13 парова конфликтних бродова, који би да заузму исту позицију у вез×вријеме координатном систему (матрици). У оптималном рјешењу, један брод у таквом конфликтном пару обично узима позицију, за коју се плаћа казна, а други брод узима позицију, за коју се не плаћа казна. То за посљедицу има стварање *„лоше“* ω пермутације у ERH, по којој даље SEDA покушава да претражи простор допустивих рјешења. Овај примјер показује и то да добра процјена вриједности функције критеријума, није довољна за ефикасан рад SEDA. У просјеку добра оцјена свакако скраћује рад SEDA, што показују просјечна времена рјешавања у прошлом потпоглављу, али нужно не значи брзо рјешавање.

Инстанцу бр. 76 SEDA+ERH+D&C је ријешио за 0.578 секунди, јер је у стању да 13 конфликтних парова бродова рјешава као одвојене проблеме, отуда и огромна разлика у временима рјешавања. У овом случају је SEDA+ERH+D&C приближно 4669 пута бржи у односу на SEDA+ERH.

Осврнимо се на неколико тешких инстанци за SEDA+ERH+D&C. DBAP инстанце бр. 100, 268 и 444 из Класе I, за 45 бродова, 5 везова и 56 временских јединица, нијесу ријешене за мање од пола сата. Инстанца бр. 100 је рјешена у другом покушају за 20 минута и 49 секунди, због тога што је у другом покушају ERH хеуристика дала боље оцјене вриједности оптимума за подпроблеме које је генерисао D&C. Инстанца бр. 268 је ријешена за нешто мање од 5 сати рада (4:48:20). Инстанца бр. 444 није ријешена ни након 12 сати рада. Један од разлога због којег су инстанце бр. 268 и бр. 444 тешки за SEDA+ERH+D&C је стварање великих подпроблема од 33 брода, као и велики број међусобно конфликтних бродова.

DBAP инстанце бр. 299 и 377 из Класе II, за 105 бродова, 8 везова и 112 временских јединица, такође нијесу ријешене за мање од пола сата. Инстанца бр. 299 ријешена је за 4 сата и 22 минута (4:22:32), док је инстанца бр. 377 у другом покушају ријешена за 11 минута и 32 секунде (11:32). Инстанца бр. 299 ствара потпроблем од 38 бродова, што га чини тешким за SEDA+ERH+D&C, док је код инстанце бр. 377 лоша процјена вриједности оптимума проузроковала не рјешавање проблема у првом покушају за мање од пола сата.

Из наведеног закључујемо да два главна разлога за споро рјешавање инстанци од стране алгоритма SEDA+ERH+D&C могу бити:

1. лоша процјена вриједности оптимума ERH, као горњег ограничења за рад SEDA и на основу тога створена *„лоша“* ω пермутација, по који SEDA одређује позицију бродова или
2. стварање великих подпроблема, који поништавају ефиксаност D&C.

У даљем развоју алгоритама базираних на *методи седиментације*, треба се концентрисати на отклањане 1. разлога спорог рјешавања, будући да на 2. разлог није могуће значајно алгоритамски утицати.

6.4.4. Поређење са *CPLEX-ом* и методе седиментације у рјешавању *Berth Allocation problem*-а

Линеарно програмирање и његове варијанте, заједно са њиховим бројним техникама оптимизације представљају један од најчешће употребљавани

метод за рјешавање проблема оптимизације. Због тога ћемо дати кратко поређење CPLEX-а, као једног од најпопуларнијих програмских пакета за рјешавање проблема линеарног програмирања, и *методе седиментације*.

Упоређење ћемо извршити на DBAP и то на првих 10 проблема Класе I, са 25 бродова. За поређење SEDA и SEDA+ERH са CPLEX-ом користили смо његову верзију 11.2, која нам је била доступна љубазношћу Математичког института САНУ. Рачунар, на којем је радио CPLEX, имао је процесор *Intel Core 2 Duo E6750 CPU @ 2.66-GHz*, меморију од 8 GB RAM и покретао га је *Linux Slackware 12 (kernel version 2.6.21.5)* оперативни систем.

Према *PassMark (www.passmark.com)* сајту, који садржи упоређења брзина са процесоре, оцјена *Intel Core 2 Duo E6750* процесора за *single thread* је 1002, док је за процесор *Intel Core i7-4500U* оцјена истог типа 1578. То значи да је *Intel Core 2 Duo E6750* приближно 0.63498 пута бржи у односу на *Intel Core i7-4500U*. Због тога множићемо CPLEX времена рјешавања са фактором 0.63498, како би добили приближно времена рјешавања као да је CPLEX радио на рачунару са *Intel Core i7-4500U* процесором.

Табела 6.4.7 *– Поређење CPLEX-а и SEDA, SEDA+ERH, SEDA+ERH+D&C.* Поређење је направљено за рјешавање DBAP-а, Класа I, са 25 бордова, 5 везова и 56 временских јединица. Времена су изражена у секундама, док је однос брзине извршавања приказан болдом.

Бр.	*Опт.*	*SEDA*	*SEDA+ ERH*	*SEDA+ ERH+ D&C*	*CPLEX*	*CPLEX equ*	$\times_{SEDA}$	$\times_{SEDA+ERH}$	$\times_{SEDA+ERH+D\&C}$
1	68	2.281	0.218	0.117	15316.500	9725.686	**4264**	**44613**	**44613**
2	24	0.125	0.035	0.129	6577.400	4176.524	**33412**	**119329**	**119329**
3	54	0.083	0.203	0.144	6187.890	3929.193	**47340**	**19356**	**19356**
4	25	0.546	0.047	0.094	4758.500	3021.557	**5534**	**64288**	**64288**
5	52	0.593	0.219	0.094	11063.300	7024.985	**11847**	**32078**	**32078**
6	21	0.047	0.047	0.101	2440.720	1549.811	**32975**	**32976**	**32976**
7	60	8.076	0.218	0.140	16385.400	10404.417	**1288**	**47727**	**47727**
8	16	0.096	0.096	0.094	4678.640	2970.847	**30946**	**30946**	**30946**
9	27	0.203	0.039	0.078	134.279	85.265	**420**	**2186**	**2186**
10	29	0.047	0.187	0.093	2764.770	1755.576	**37353**	**9389**	**9389**
Σ		**12.097**	**1.309**	**1.084**	**70307.399**	**44643.861**	**3690**	**34105**	**41184**

Табела 6.4.7. садржи поређење времена рјешавања првих 10 инстанци DBAP-а, Класа I проблема са 25 бродова, 5 везова и 56 временских јединица. У првој колони и број проблема, а у другој вриједност оптимума. Затим редом иду времена рјешавања метода: SEDA, SEDA+ERH, SEDA+ERH+D&C и CPLEX-а. Потом иде колона CPLEX equ, која нам прерачунава колико би било еквивалентно вријеме рјешавања, да је CPLEX радио на *Intel Core i7-4500U* процесору, као што је то био случај са алгоритмима базираним на *методи седиментације*. У посљедње три колоне су односи брзина израчунавања CPLEX-а и алгоритама базираним на *методи седиментације*. На крају, у посљедњем реду дати су сумарни подаци за сваку колону. Колико су алгоритми базирани на *методи седиментације* супериорни у односу на CPLEX довољно је обратити пажњу на те сумарне податке. SEDA је **3690** пута бржи, SEDA+ERH је **34105** пута бржи и коначно SEDA+ERH+D&C је **41184** пута бржи.

Тестови на CPLEX-у вршени на верзији 11.2. Тренутно посљедња расположива верзија CPLEX-а је 12.6.1. Према страници *IBM ILOG CPLEX Optimizer page*[6], верзија CPLEX-а 12.6.1 је до највише **40** пута бржа у односу на верзију 11.2. Узмемо ли да је овај податак тачан, из њега закључујемо да је SEDA бар **92.25** пута бржа, SEDA+ERH је бар **852.63** и SEDA+ERH+D&C је бар **1029.60** пута бржа у односу на CPLEX 12.6.1.

Из свега наведеном закључујемо да су алгоритми базирани на *методи седиментације* супериорнији од CPLEX-а, било да се ради о његовој верзији 11.2 или верзији 12.6.1. Цјелобројно линеарни модел BAP-а, коришћен за ове тестове на CPLEX-у, је сувише комплексан да би могао да ријеши проблеме са 25 или више бродова, нашто упућује (Davidović, et al., 2012). Исто важи и за примјену разних хеуристика цјелобројног линеарног програмирања на овај модел BAP-а.

[6] http://www-01.ibm.com/software/commerce/optimization/cplex-performance/#improvements

6.4.5. Поређење са *state of the art* алгоритмима за рјешавање *Berth Allocation problem*-а

Најзначајнији скорији приступ рјешавању BAP налазимо у (Vacca, Salani, & Bierlaire, 2011). Аутори, као што смо то већ раније навели, разматрају проблем *Tactical Berth Allocation Problem* (TBAP), којег су увели (Giallombardo, Moccia, Salani, & Vacca, 2010). Ова верзија TBAP-а врши додјелу, како везова тако и кранова, бродовима које треба опслужити у луци. Кранови се додјељују по већ унапријед утврђеним распоредима, који се називају профили. Функција критеријума максимизује вриједност профила и трошкове одржавања створене додјелом веза. Са друге стране, SEDA, SEDA+ERH и SEDA+ERH+D&C алгоритми, овдје изложени, базирају се на Рашиди-Цанг моделу (Rashidi & Tsang, 2013), чија функција критеријума минимализује казну мјеста веза, чекања брода, пожуривања брода и кашњења брода. Ове двије чињенице:

1. разлика у формулацији проблема BAP, у односу на BACAP и
2. разлика функције критеријума;

чине директно поређење ова два метода немогућим. Ова два приступа направљена су тако да рјешавању два различита типа BAP. Због тога директно поређење времена рјешавања није примјерено. Оно што подразумијевамо под BAP-ом није један проблем, већ цијела фамилија проблема. Уколико се не ради о истој врсти BAP-а, тада поређење није одговарајуће. Колико познато, SEDA, SEDA+ERH и SEDA+ERH+D&C су прве имплементације које рјешавају DBAP и HBAP на основу Рашиди-Цанг модела.

Умјесто поређења ова два метода, приказаћемо просјечна времена рјешавања изражена у секундама у табели 6.4.8 за димензије које се разматрају и рјешавају у (Giallombardo, Moccia, Salani, & Vacca, 2010) и (Vacca, Salani, & Bierlaire, 2011).

Табела 6.4.8 *– Димензије проблема DABP-а који се рјешавају у (Giallombardo, Moccia, Salani, & Vacca, 2010) и (Vacca, Salani, & Bierlaire, 2011)*

Класа и тип	***Број везова*** *m*	***Хоризонт планирања*** *T*	***Број бодова*** *l*	***Просјечно вријеме рјешавања*** *SEDA+ERH*	***Просјечно вријеме рјешавања*** *SEDA+ERH*
Класа I, DBAP	5	56	30	1.33	0.39
Класа II, DBAP	8	112	50	1.52	0.08

Остали наведени приступи који траже оптимално рјешење BAP-а разматрају, или специјалне верзије BAP, као (Hendriks, Armbruster, Lefeber, & Udding, 2012), (Chen, Lee, & Cao, 2012) и (Hu, Hu, & Du, 2014), или као (Umang, Bierlaire, & Vacca, 2013) и (Robenek, Umang, Bierlaire, & Ropke, 2014) разматрају проблем BAP-а у лукама за расуте терете. Свима је заједничко да не разматрају број бродова већи него што SEDA, SEDA+ERH и SEDA+ERH+D&C могу успјешно да ријеше.

Слично је и са хеуристичким и мета хеуристичким приступима рјешавању DBAP-а и HBAP-а. У прегледу који налазимо у (Kovač, Metaheuristic approaches for the Berth Allocation Problem, 2016), у већини приступа највећи број бродова за које се рјешава БАП је 100, са изузетком (Hansen, Oguz, & Mladenović, 2008) и (Cheong, Tan, Liu, & Lin, 2010), који рјешавају проблеме и до 200 бродова. Број везова варира од 2 до 13 за DBAP и од 5 до 30 за HBAP. Хоризонт планирања варира, како у временској дужини, тако и у броју временских јединица. Најчешће је период од 1 до 2 седмице, мада има и других. Ове податке наводимо само ради сагледавања димензија проблем који рјешавају методи са хеуристичким и мета хеуристичким приступом и њиховог упоређења са димензијама које могу да ријеше алгоритми базирани на *методи седиментације*. Њихов преглед дат је на илустрацији 6.4.5.

Из свега наведеног закључујемо да алгоритми базирани на *методи седиментације*: SEDA, SEDA+ERH, SEDA+ERH+D&C могу рјешавати проблеме DBAP и HBAP веома ефикасно, што потврђују тест примјери. Не само да могу успјешно рјешавати проблеме са истим димензијама као тренутни *state of art* приступи, већ и значајно повећавајући димензије проблема који су тачно рјешиви. За Класу I, DBAP-а у стању су да ријеше проблеме са 40 бродова, за 10 више, а за Класу II, DBAP-а рјешавају проблеме са 100 бродова, за 50 више, него што су то разматрали ранији тачни рачни приступи рјешавању BAP-а. Напоменимо да су за ове димензије проблема просјечне брзине тачног рјешавања испод 10 секунди.

7. ЗАКЉУЧНА РАЗМАТРАЊА

Метод седиментације могуће је примијенити на широк скуп ПО. Питање моделовања проблема за рјешавање је кључно за ефикасност рјешавања. Два различита моделовања истог проблема могу се значајно разликовати у ефикасности, што показује примјер Max-SAT проблема. Уколико моделовање успије да укључи поред самог SEDA и његове двије хеуристике ERH и D&C, утолико ће ефикасност рјешавања бити већа, што потврђује анализа резултата BAP-а.

Имплементација *методе седиментације* спада у средње захтијевне. Поред програмирања самог контролног механизма методе, потребно је још доста програмирања за одржавања структура података, које су потребне за рад метода.

У наведеним примјерима најбоље се показала варијанта SEDA+ERH+D&C *методе седиментације,* што је очекивано, обзиром да она садржи највећи степен оптимизацију у поређењу са осталим приказаним варијантама *методе седиментације.*

Проблем који је најуспјешније ријешен *методом седиментације* је BAP. Резултати су на нивоу тренутних *state of art* приступа.

Даљи рад ће бити усмјерен ка даљој оптимизацији *методе седиментације*. Хеуристику *„процијени и преуреди“* могуће је усавршити напреднијим алгоритмом, који ће долазити до боље оцјене оптималног рјешења и до бољег поретка, по којој основни алгоритам *методе седиментације* разматра промјењиве одлучивања. Такође, могуће је хеуристику *„подијели, па владај“* јаче повезати са основним алгоритмом *методе седиментације* и на тај начин побољшати његову ефикасност.

ЛИТЕРАТУРА

Ansotegui, C., Bonet, M. L., & Levy, J. (2013). SAT-based MaxSAT algorithms. *Artificial Inteligence, 196*, 77-105.

Biere, A., Heule, M., Van Maaren, H., & Walsh, T. (Уредници). (2009). *Handbook of Satisfiability*. Amsterdam: IOS Press.

Bierwirth, C., & Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research, 202*(3), 615-627.

Bierwirth, C., & Meisel, F. (2015). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research, 244*(3), 675-689.

Chen, C. Y., & Hsieh, T. W. (1998). A time-space network model for the berth allocation problem. *Proceedings of 19th IFIP TC7 Conference on System Modelling and Optimization*. Cambridge.

Chen, J. H., Lee, D.-H., & Cao, J. X. (2012). A combinatorial benders' cuts algorithm for the quayside operation problem at container terminals. *Transportation Research Part E, 48*, 266-275.

Cheong, C. Y., Tan, K. C., Liu, D. K., & Lin, C. J. (2010). Multi-objective and prioritized berth allocation in container ports. *Annals of Operations Research, 180*(1), 63-103.

Conway, J. H., & Guy, R. K. (1996). *The Book of Numbers*. New York: Copernicus.

Cook, S. A. (1970). The complexity of theorem-proving procedures. *3rd STOC*, (стр. 151-158). New York.

Cordeau, J. F., Laporte, G., Legato, P., & Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem. *Transportation Science, 39*(4), 526-538.

Cvetković, D., Čangalović, M., Dugošija, Đ., Kovačević-Vujičić, V., Simić, S., & Vuleta, J. (1996). *Kombinatorna optimizacija: Matematička teorija i algoritmi*. Beograd: Društvo operacionih istraživača Jugoslavije.

Dai, J., Lin, W., Moorthy, R., & Teo, C.-P. (2008). Berth Allocation Planning Optimization in Container Terminals. У *Supply Chain Analysis* (Т. 119, стр. 69-104). New York: Springer.

Davidović, T., Kovač, N., & Stanimirović, Z. (2015). VNS-based approach to minimum cost hybrid berth allocation problem. *Proceedings SYM-OP-IS 2015: XLII International Symposium on Operations Research.* Ivanjica.

Davidović, T., Lazić, J., Mladenović, N., Kordić, S., Kovač, N., & Dragović, B. (2012). MIP-Heuristics for Minimum Cost Berth Allocation Problem. *International Conference on Traffic and Transort Engineering (ICTTE 2012)*, (стр. 21-28). Belgrade.

Davis, M., & Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the ACM, 7*(3), 201-215.

Davis, M., Logemann, G., & Loveland, D. (1962). A machine program for theorem proving. *Journalof the ACM, 5*, 394-397.

de Oliveira, R. M., Mauri, G. R., & Lorena, L. N. (2012). Clustering Search for the Berth Allocation Problem. *Expert Systems with Applications, 39*(5), 5499-5505.

Dechter, R. (2003). *Constraint Processing.* San Francisco: Morgan Kaufmann Publishers.

Freuder, E. C., & Mackworth, A. K. (2006). Constraint Satisfaction: An Emerging Paradigm. In F. Rossi, P. Van, & T. Walsh (Eds.), *Handbook of Constraint Programming.* Amsterdan: Elsevier.

Fu, Y. M., & Diabat, A. (2015). A Lagrangian relaxation approach for solving the integrated quay crane assignment and scheduling problem. *Applied Mathematical Modelling, 39*, 1194-1201.

Fu, Z. (2009). *Expanding the power of boolean satisfiability: Techniques and applications, Ph.D. thesis.* Princeton: Prinston University.

Giallombardo, G., Moccia, L., Salani, M., & Vacca, I. (2010). Modeling and solving the tactical berth allocation problem. *Transportation Research B, 44*(3), 400-415.

Han, M., & Sun, J. L. (2006). The algorithm for berth scheduling problem by the hybrid optimization strategy GASEDA. *Proceedings of the 9th*

International Conference of Control, Automation, Robotics and Vision, ICARCV '06 (стр. 1-4). Washington: IEEE Computer Society.

Hansen, P., Oguz, C., & Mladenović, N. (2008). Variable neighbourhood search for minimum cost berth allocation. *European Journal of Operational Research, 191*(3), 636-649.

Haralic, R. M., Miasnikov, A. D., & Myasnikov, A. G. (2005). Heuristics for the Whitehead Minimization Problem. *Experimental Mathematics, 14*, 7-14.

Hendriks, M., Armbruster, D., Lefeber, E., & Udding, J. (2012). Strategic allocation of cyclically vessels for multi-terminal container operators. *Flexibile Services and Manufacturing Journal, 24*(3), 248-273.

Heras, F., Larrosa, J., & Oliveras, A. (2007). MiniMaxSat: A new weighted Max-SAT solver. *Proceedings of the 10th International Conference on Theory and Application of Satisfiability Testing SAT'07*, (стр. 41-55).

Hu, Q.-M., Hu, Z.-H., & Du, Y. (2014). Berth and quay-crane allocation problem considering fuel consumption and emissions from vessels. *Computers & Industrial Engineering, 70*, 1-10.

Imai, A., Nishimura, E., & Papadimitriou, S. (2001). The dynamic berth allocation problem for container port. *Transportation Research Part B, 35*(4), 401-417.

Imai, A., Nishimura, E., & Papadimitriou, S. (2008). Berthing ships at a multi-user container terminal with a limited quay capacity. *Transportation Research Part E, 179*(2), 579-593.

Imai, A., Nishimura, E., & Papadimitriou, S. (2013). Marine container terminal configurations for efficient handling of mega-containerships. *Transportation Research Part E, 49*, 141-158.

Imai, A., Nishimura, E., Hattori, M., & Papadimitriou, S. (2007). Berth allocation at indented berths for mega-containerships. *European Journal of Operational Research, 179*(2), 579-593.

Knuth, D. E. (2016). *The Art of Computer Programming, Combinatorial Algorithms: Part 2* (Том. Volume 4,). Преузето са http://www.cs.utsa.edu/~wagner/knuth/fasc5b2016.05.25.pdf

Kordić, S. (2016). Application of Sedimentation Algorithm for Solving Max-SAT Problem. *Mathematica Montisnigri*.

Kordić, S., Davidović, T., Kovač, N., & Dragović, B. (2016). Combinatorial Approach to Exactly Solving Discrete and Hybrid Berth Allocation Problem. *Applied Mathematical Modelling*. doi:10.1016/j.apm.2016.05.004

Kordić, S., Kovač, N., & Davidović, T. (2015). Divide and Conquer Approach to Discrete Berth Allocation Problem. *The 12th Balkan Conference on Operational Research BALCOR 2015, Volume: "Mircea cel Batran" Naval Academy Scientific Bulletin*, *XVIII (2)*, pp. 307-316. Constanta.

Kovač, N. (2013). Bee Colony Optimization Algorithm for the Minimum Cost Berth Allocation Problem. *Proceedings of the XI Balkan Conference on Operational Research, BALCOR 2013*, (стр. 245-254). Belgrade-Zlatibor.

Kovač, N. (2016). Metaheuristic approaches for the Berth Allocation Problem. *(submitted for printing)*.

Kubiak, R., Rudzinski, R., & Sokolowski, S. (1991). *An Introduction to Programming with Specifications.* London: Academic Press.

Land, A. H., & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica, 28*(3), 497-520. doi:10.2307/1910129

Li, C. M., Manya, F., Mohamedou, N. O., & Planes, J. (2009). Exploiting cycle structures in Max-SAT. *Proceedings of the 12th International Conference on Theory and Application of Satisfiability Testing SAT'09*, (стр. 467-480).

Lim, A. (1998). The berthing planning problem. *Operational Research Letters, 22*(2), 105-110.

Lin, H., Su, K., & Li, C. M. (2008). Within-problem learning for efficient lower bond computation in Max-SAT solving. *Proceedings of the 23rd National Conference on Artificial Inteligence AAAI'08*, (стр. 351-358).

Lin, S. W., Ying, K. C., & Wan, S. Y. (2014). Minimizing the Total Service Time of Discrete Dynamic Berth Allocation Problem by an Iterated Greedy Heuristic. *The Scientific World Journal*. doi:10.1155/2014/218925

Little, J. D., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An Algorithm for the Traveling Salesman Problem. *Operations Research, 11*(6), 972-989.

Lyndon, R., & Schupp, P. (1977). *Combinatorial Group Theory, Series of Modern Studies in Mathematics* (T. 89). Berlin: Springer-Verlag.

Marić, F. (2009). Formalization and Implementation of Modern SAT Solvers. *Journal of Automated Reasoning, 43*(1), 81-119.

Mauri, G. R., Oliveira, A. M., & Lorena, A. N. (2008). A hybrid column generation approach for the bearth allocation problem. *Lecture Notes in Computer Science, 4972*, 110-122.

Meisel, F. (2009). *Seaside Operations Planning in Container Terminals.* Berlin: Physica Verlag.

Meisel, F., & Bierwirth, C. (2013). A framework for integrated berth allocation and crane planning in seaport container terminals. *Transportation Science, 47*(2), 131-147.

Monaco, F. M., & Samara, M. (2009). The berth allocation problem: a strong formulation solved by a lagrangian approach. *Transportation Science, 41*(2), 256-280.

Moorthy, R., & Teo, C. P. (2006). Berth management in container terminal: the template design problem. *OR Spectrum, 28*(4), 495-518.

Myasnikov, A. D., & Haralic, R. M. (2006). A hybrid search algorithm for the Whitehead Minimization problem . *Journal of Symbolic Computation, 41*, 818-834.

Nishimura, E., Imai, A., & Papadimitriou, S. (2001). Berth allocation planning in the public berth system by genetic algorithms. *European Journal of Operational Research, 131*(2), 282-292.

Rashidi, H., & Tsang, E. K. (2013). Novel constrains satisfaction models for optimization problems in container terminals. *Applied Mathematical Modelling, 37*, 3601-3634.

Robenek, T., Umang, N., Bierlaire, M., & Ropke, S. (2014). A branch-and-price algorithm to solve the integrated berth allocation and yard assignment problem in bulk ports. *European Journal of Operational Research, 235*, 399-411.

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Harlow: Pearson Education Limited.

Tsang, E. (1993). *Foundation of Constraint Satisfaction*. London: Academic Press.

Umang, N., Bierlaire, M., & Vacca, I. (2013). Exact and heuristic methods to solve berth allocation problems in bulk ports. *Transportation Research Part E, 54*, 14-31.

Vacca, I., Salani, M., & Bierlaire, M. (2011). *An exact algorithm for integrated planning of berth allocation and quay crane assignment*. Lausanne: Ecole Polytechnique Federale de Lausanne.

Vujošević, M. (2012). *Metode optimizacije u inženjerskom menadžmentu*. Beograd: Akademija inženjerskih nauka Srbije.

Whitehead, J. C. (1936). On equivalent sets of elements in a free group. *Annals of Mathematics, 37*, 782-800.

Zhen, P., Lee, L. H., & Chew, E. P. (2011). A decision model for berth allocation under uncertainty. *European Journal of Operational Research, 212*(1), 54-68.

Zhou, P., Kang, H., & Lin, L. (2006). A Dynamic Berth Allocation Model Based on Stochastic Consideration. *Proceeding of the 6th World Congress on Intelligent Control and Automation. 2*, стр. 7297-7301. Washington: IEEE Computer Society.

Петровић, З., & Мијајловић, Ж. (2012). *Математичка логика - елементи теорије скупова*. Београд: Завод за уџбенике.

ПРИЛОЗИ

SEDA

рјешавање инстанце бр. 2, DBAP, Каса I

5 везова, **56** временских јединица и **35** бродова
Оптимум: **59**, вријеме рјешавања: **13.739** s.

```
PROBLEM No. 7

Input problem parameters:
```

l	EST	ETA	a	b	d	LDT	s	C1	C2	C3	C4
1	1	51	1	1	52	56	1	2	3	3	9
2	1	46	6	1	52	56	2	4	9	9	27
3	1	19	2	1	21	56	3	2	3	3	9
4	1	34	5	1	39	56	4	3	6	6	18
5	1	1	1	1	2	56	5	2	3	3	9
6	1	2	1	1	3	56	1	2	3	3	9
7	1	50	4	1	54	56	2	3	6	6	18
8	1	35	3	1	38	56	3	2	3	3	9
9	1	17	8	1	25	56	4	4	9	9	27
10	1	40	7	1	47	56	5	4	9	9	27
11	1	37	1	1	38	56	1	2	3	3	9
12	1	31	2	1	33	56	2	2	3	3	9
13	1	10	5	1	15	56	3	3	6	6	18
14	1	22	4	1	26	56	4	3	6	6	18
15	1	8	4	1	12	56	5	3	6	6	18
16	1	44	2	1	46	56	1	2	3	3	9
17	1	50	2	1	52	56	2	2	3	3	9
18	1	13	1	1	14	56	3	2	3	3	9
19	1	42	2	1	44	56	4	2	3	3	9
20	1	23	4	1	27	56	5	3	6	6	18
21	1	19	2	1	21	56	1	2	3	3	9
22	1	26	5	1	31	56	2	3	6	6	18
23	1	1	5	1	6	56	3	3	6	6	18
24	1	4	5	1	9	56	4	3	6	6	18
25	1	29	5	1	34	56	5	3	6	6	18
26	1	32	2	1	34	56	1	2	3	3	9
27	1	24	1	1	25	56	2	2	3	3	9
28	1	36	4	1	40	56	3	3	6	6	18
29	1	5	2	1	7	56	4	2	3	3	9
30	1	5	4	1	9	56	5	3	6	6	18
31	1	20	7	1	27	56	1	4	9	9	27
32	1	7	5	1	12	56	2	3	6	6	18
33	1	20	2	1	22	56	3	2	3	3	9
34	1	54	1	1	55	56	4	2	3	3	9
35	1	48	8	1	56	56	5	4	9	9	27

```
SA Solver

Rearranged set:

  1: 0.000000    2: 0.000000    3: 0.000000    4: 0.000000    5: 0.000000
  6: 0.000000    7: 0.000000    8: 0.000000    9: 0.000000   10: 0.000000
 11: 0.000000   12: 0.000000   13: 0.000000   14: 0.000000   15: 0.000000
 16: 0.000000   17: 0.000000   18: 0.000000   19: 0.000000   20: 0.000000
 21: 0.000000   22: 0.000000   23: 0.000000   24: 0.000000   25: 0.000000
 26: 0.000000   27: 0.000000   28: 0.000000   29: 0.000000   30: 0.000000
 31: 0.000000   32: 0.000000   33: 0.000000   34: 0.000000   35: 0.000000
```

```
Solutions for 35 variables starting at 0.015 sec:

 1   :    164          0.015 sec
 2   :    161          0.015 sec
 3   :    123          0.015 sec
 4   :     90          0.015 sec
 5   :     66          0.031 sec
 6   :     60          0.047 sec
 7   :     59          0.416 sec

Optimal solution of the problem:

     l |  berth  time     f
-------------------------------
     1 |      1|   51|    0
     2 |      2|   46|    0
     3 |      3|   18|    3
     4 |      4|   34|    0
     5 |      5|    1|    0
     6 |      1|    2|    0
     7 |      3|   50|   12
     8 |      3|   33|    6
     9 |      4|   17|    0
    10 |      5|   40|    0
    11 |      1|   37|    0
    12 |      2|   31|    0
    13 |      3|   10|    0
    14 |      3|   22|   12
    15 |      5|    8|    0
    16 |      1|   44|    0
    17 |      1|   49|    7
    18 |      2|   13|    2
    19 |      4|   42|    0
    20 |      5|   23|    0
    21 |      1|   18|    3
    22 |      2|   26|    0
    23 |      3|    1|    0
    24 |      4|    4|    0
    25 |      5|   29|    0
    26 |      1|   32|    0
    27 |      2|   24|    0
    28 |      3|   36|    0
    29 |      2|    5|    8
    30 |      5|    4|    6
    31 |      1|   20|    0
    32 |      2|    7|    0
    33 |      3|   20|    0
    34 |      4|   54|    0
    35 |      5|   48|    0

Optimal solution objective function value: 59

Time for solving the problem: 13.737 sec
```

SEDA+ERH
рјешавање инстанце бр. 27, DBAP, Каса I

5 везова, **56** временских јединица и **40** бродова.
Оптимум: **125**, вријеме рјешавања: **47.014** s.

```
PROBLEM No. 27


Input problem parameters:
```

l	EST	ETA	a	b	d	LDT	s	C1	C2	C3	C4
1	1	14	4	1	18	56	1	3	6	6	18
2	1	47	4	1	51	56	2	3	6	6	18
3	1	53	3	1	56	56	3	2	3	3	9
4	1	25	1	1	26	56	4	2	3	3	9
5	1	24	3	1	27	56	5	2	3	3	9
6	1	13	8	1	21	56	1	4	9	9	27
7	1	32	2	1	34	56	2	2	3	3	9
8	1	36	7	1	43	56	3	4	9	9	27
9	1	7	1	1	8	56	4	2	3	3	9
10	1	31	1	1	32	56	5	2	3	3	9
11	1	19	3	1	22	56	1	2	3	3	9
12	1	43	8	1	51	56	2	4	9	9	27
13	1	30	2	1	32	56	3	2	3	3	9
14	1	2	6	1	8	56	4	4	9	9	27
15	1	27	5	1	32	56	5	3	6	6	18
16	1	42	1	1	43	56	1	2	3	3	9
17	1	24	2	1	26	56	2	2	3	3	9
18	1	7	5	1	12	56	3	3	6	6	18
19	1	15	7	1	22	56	4	4	9	9	27
20	1	42	1	1	43	56	5	2	3	3	9
21	1	54	2	1	56	56	1	2	3	3	9
22	1	27	5	1	32	56	2	3	6	6	18
23	1	29	5	1	34	56	3	3	6	6	18
24	1	27	2	1	29	56	4	2	3	3	9
25	1	34	7	1	41	56	5	4	9	9	27
26	1	2	8	1	10	56	1	4	9	9	27
27	1	43	2	1	45	56	2	2	3	3	9
28	1	51	1	1	52	56	3	2	3	3	9
29	1	33	3	1	36	56	4	2	3	3	9
30	1	49	2	1	51	56	5	2	3	3	9
31	1	53	1	1	54	56	1	2	3	3	9
32	1	32	5	1	37	56	2	3	6	6	18
33	1	47	3	1	50	56	3	2	3	3	9
34	1	5	6	1	11	56	4	4	9	9	27
35	1	34	3	1	37	56	5	2	3	3	9
36	1	19	2	1	21	56	1	2	3	3	9
37	1	25	1	1	26	56	2	2	3	3	9
38	1	10	5	1	15	56	3	3	6	6	18
39	1	34	3	1	37	56	4	2	3	3	9
40	1	53	2	1	55	56	5	2	3	3	9

```
SA + ERH1 Limited Solver

Number of estimations: 90  -  Number of nodes: 1200

Estimations for 40 variables starting at 0.016 sec:

 1   :    213      not 0:  14       0.016 sec
 2   :    210      not 0:  15       0.016 sec
 3   :    207      not 0:  14       0.078 sec
 4   :    167      not 0:  16       0.100 sec
```

```
 5   :    160      not 0:  16       0.100 sec
 6   :    156      not 0:  16       0.100 sec
 7   :    154      not 0:  16       0.132 sec
 8   :    152      not 0:  16       0.132 sec
 9   :    148      not 0:  17       0.363 sec
10   :    147      not 0:  17       0.363 sec

Rearranged set:

 14: 0.162162    6: 0.121622   18: 0.101351    2: 0.081081    1: 0.081081
 29: 0.081081   39: 0.060811   13: 0.054054   15: 0.040541   35: 0.040541
 10: 0.033784   36: 0.027027    7: 0.027027   27: 0.027027    5: 0.020270
 17: 0.020270    9: 0.013514   26: 0.000000   12: 0.000000   25: 0.000000
 19: 0.000000    8: 0.000000   34: 0.000000   23: 0.000000   22: 0.000000
 38: 0.000000   32: 0.000000    3: 0.000000   11: 0.000000   33: 0.000000
 24: 0.000000   21: 0.000000   30: 0.000000   40: 0.000000   16: 0.000000
 28: 0.000000   37: 0.000000   31: 0.000000   20: 0.000000    4: 0.000000


Solutions for 40 variables starting at 0.564 sec:

11   :    146          0.702 sec
12   :    144          0.702 sec
13   :    142          1.504 sec
14   :    140          1.504 sec
15   :    138          1.504 sec
16   :    136          1.504 sec
17   :    135          1.519 sec
18   :    133          1.519 sec
19   :    132          1.804 sec
20   :    130          1.804 sec
21   :    128          1.820 sec
22   :    127          5.023 sec
23   :    125          5.038 sec

Optimal solution of the problem:

     l |  berth  time     f
-------------------------------
     1 |      2|   14|   12
     2 |      1|   47|   12
     3 |      3|   53|    0
     4 |      4|   25|    0
     5 |      5|   23|    3
     6 |      1|   13|    0
     7 |      1|   32|    4
     8 |      3|   36|    0
     9 |      2|    7|    4
    10 |      4|   30|    5
    11 |      2|   19|    6
    12 |      2|   43|    0
    13 |      1|   30|    8
    14 |      4|    2|    0
    15 |      5|   26|    6
    16 |      1|   42|    0
    17 |      2|   24|    0
    18 |      3|    5|   12
    19 |      4|   15|    0
    20 |      5|   42|    0
    21 |      1|   54|    0
    22 |      2|   27|    0
    23 |      3|   29|    0
    24 |      4|   27|    0
    25 |      5|   34|    0
    26 |      1|    2|    0
```

```
    27 |      3|   43|    4
    28 |      3|   51|    0
    29 |      4|   31|    6
    30 |      5|   49|    0
    31 |      1|   53|    0
    32 |      2|   32|    0
    33 |      3|   47|    0
    34 |      5|    5|   24
    35 |      5|   31|    9
    36 |      3|   19|    8
    37 |      1|   25|    2
    38 |      3|   10|    0
    39 |      4|   34|    0
    40 |      5|   53|    0

Optimal solution objective function value: 125

Time for solving the problem: 47.014 sec
```

SEDA+ERH+D&C
рјешавање инстанце бр. 2, DBAP, Каса I

5 везова, **56** временских јединица и **40** бродова.
Оптимум: **125**, вријеме рјешавања: **0.400** s.

```
PROBLEM No. 27


Input problem parameters:
```

l	EST	ETA	a	b	d	LDT	s	C1	C2	C3	C4
1	1	14	4	1	18	56	1	3	6	6	18
2	1	47	4	1	51	56	2	3	6	6	18
3	1	53	3	1	56	56	3	2	3	3	9
4	1	25	1	1	26	56	4	2	3	3	9
5	1	24	3	1	27	56	5	2	3	3	9
6	1	13	8	1	21	56	1	4	9	9	27
7	1	32	2	1	34	56	2	2	3	3	9
8	1	36	7	1	43	56	3	4	9	9	27
9	1	7	1	1	8	56	4	2	3	3	9
10	1	31	1	1	32	56	5	2	3	3	9
11	1	19	3	1	22	56	1	2	3	3	9
12	1	43	8	1	51	56	2	4	9	9	27
13	1	30	2	1	32	56	3	2	3	3	9
14	1	2	6	1	8	56	4	4	9	9	27
15	1	27	5	1	32	56	5	3	6	6	18
16	1	42	1	1	43	56	1	2	3	3	9
17	1	24	2	1	26	56	2	2	3	3	9
18	1	7	5	1	12	56	3	3	6	6	18
19	1	15	7	1	22	56	4	4	9	9	27
20	1	42	1	1	43	56	5	2	3	3	9
21	1	54	2	1	56	56	1	2	3	3	9
22	1	27	5	1	32	56	2	3	6	6	18
23	1	29	5	1	34	56	3	3	6	6	18
24	1	27	2	1	29	56	4	2	3	3	9
25	1	34	7	1	41	56	5	4	9	9	27
26	1	2	8	1	10	56	1	4	9	9	27
27	1	43	2	1	45	56	2	2	3	3	9
28	1	51	1	1	52	56	3	2	3	3	9
29	1	33	3	1	36	56	4	2	3	3	9
30	1	49	2	1	51	56	5	2	3	3	9
31	1	53	1	1	54	56	1	2	3	3	9
32	1	32	5	1	37	56	2	3	6	6	18
33	1	47	3	1	50	56	3	2	3	3	9
34	1	5	6	1	11	56	4	4	9	9	27
35	1	34	3	1	37	56	5	2	3	3	9
36	1	19	2	1	21	56	1	2	3	3	9
37	1	25	1	1	26	56	2	2	3	3	9
38	1	10	5	1	15	56	3	3	6	6	18
39	1	34	3	1	37	56	4	2	3	3	9
40	1	53	2	1	55	56	5	2	3	3	9

```
Fragment SA + ERH Solver

Number of estimations: 36  -  Number of nodes: 80

Initial estimations for 40 variables starting at 0.016 sec:

Initial upper limit of the optimal solution: 141
```

```
Fragment level 0
     1 |  1| :  26
     2 |  1| :  19
     3 |  1| :   8
     4 |  1| :  22
     5 |  1| :  33
     6 |  1| :   5
     7 |  1| :   3
     8 |  1| :  24
     9 |  1| :  40
    10 |  1| :  21
    11 |  1| :  30
    12 |  1| :  20
    13 |  1| :  31
    14 |  1| :   4
    15 |  1| :  16
    16 |  1| :  28
    17 |  2| :  15, 10
    18 |  2| :  18, 38
    19 |  2| :  29, 39
    20 |  2| :  23, 13
    21 |  2| :  17, 37
    22 |  2| :  35, 25
    23 |  2| :  32,  7
    24 |  3| :  12,  2, 27
    25 |  3| :  34, 14,  9
    26 |  4| :   6,  1, 11, 36

--------------------------------------------------
Solving fragment 17 | l = 2

No estimations!

 15   10

 3   :      2          0.262 sec

Optimal solution of the fragmet: 2
--------------------------------------------------
Solving fragment 18 | l = 2

No estimations!

 18   38

 3   :     15          0.281 sec
 4   :     12          0.284 sec

Optimal solution of the fragmet: 12
--------------------------------------------------
Solving fragment 19 | l = 2

No estimations!

 29   39

 3   :      6          0.292 sec

Optimal solution of the fragmet: 6
Joining conflicting fragments 19 & 22
--------------------------------------------------
Solving fragment 19 | l = 4

No estimations!
```

```
 29   39   35   25

 3   :     15          0.300 sec

Optimal solution of the fragmet: 15
Joining conflicting fragments 19 & 3
Joining conflicting fragments 18 & 16
--------------------------------------------------
Solving fragment 17 | l = 7

No estimations!

 29   39   35   25    8   15   10

 3   :     26          0.300 sec
 4   :     25          0.300 sec
 5   :     23          0.300 sec

Optimal solution of the fragmet: 23
Joining conflicting fragments 17 & 5
Joining conflicting fragments 16 & 17
--------------------------------------------------
Solving fragment 16 | l = 10

Number of estimations: 1500  -  Number of nodes: 40

Estimations for 10 variables starting at 0.315 sec:

Upper limit: 39

 35   29   23   15   39    5   13   10    8   25

 4   :     33          0.315 sec

Optimal solution of the fragmet: 33
Joining conflicting fragments 16 & 3
Joining conflicting fragments 15 & 17
--------------------------------------------------
Solving fragment 15 | l = 13

Number of estimations: 1725  -  Number of nodes: 52

Estimations for 13 variables starting at 0.315 sec:

Upper limit: 46

 35   29   23   15   22   32   39    5   13    7
 10   25    8

 4   :     44          0.331 sec
 5   :     43          0.331 sec
 6   :     41          0.331 sec

Optimal solution of the fragmet: 41
--------------------------------------------------
Solving fragment 16 | l = 2

No estimations!

 17   37

 3   :      2          0.331 sec

Optimal solution of the fragmet: 2
--------------------------------------------------
```

```
Solving fragment 17 | l = 3

No estimations!

 12    2   27

 3   :     16          0.331 sec

Optimal solution of the fragmet: 16
-----------------------------------------------------
Solving fragment 18 | l = 3

No estimations!

 34   14    9

 3   :     26          0.347 sec

Optimal solution of the fragmet: 26
Joining conflicting fragments 18 & 14
-----------------------------------------------------
Solving fragment 17 | l = 5

No estimations!

 34   14    9   18   38

 3   :     41          0.347 sec
 4   :     40          0.347 sec

Optimal solution of the fragmet: 40
-----------------------------------------------------
Solving fragment 18 | l = 4

No estimations!

  6    1   11   36

 3   :     26          0.362 sec

Optimal solution of the fragmet: 26
-----------------------------------------------------

Fragment level 0
     1 |  1| :  26
     2 |  1| :  19
     3 |  1| :  33
     4 |  1| :   3
     5 |  1| :  24
     6 |  1| :  40
     7 |  1| :  21
     8 |  1| :  30
     9 |  1| :  20
    10 |  1| :  31
    11 |  1| :   4
    12 |  1| :  16
    13 |  1| :  28
    14 | 13| :  29, 39, 35, 25,  8, 15, 10,  5, 23, 13
                22, 32,  7
    15 |  2| :  17, 37
    16 |  3| :  12,  2, 27
    17 |  5| :  34, 14,  9, 18, 38
    18 |  4| :   6,  1, 11, 36
```

Optimal solution of the problem:

l	berth	time	f
1	2	14	12
2	1	47	12
3	3	53	0
4	4	25	0
5	5	23	3
6	1	13	0
7	1	32	4
8	3	36	0
9	2	7	4
10	4	30	5
11	2	19	6
12	2	43	0
13	1	30	8
14	5	2	24
15	5	26	6
16	1	42	0
17	2	24	0
18	3	5	12
19	4	15	0
20	5	42	0
21	1	54	0
22	2	27	0
23	3	29	0
24	4	27	0
25	5	34	0
26	1	2	0
27	3	43	4
28	3	51	0
29	4	31	6
30	5	49	0
31	1	53	0
32	2	32	0
33	3	47	0
34	4	5	0
35	5	31	9
36	3	19	8
37	1	25	2
38	3	10	0
39	4	34	0
40	5	53	0

Optimal solution objective function value: 125

Time for solving the problem: 0.400 sec

БИОГРАФИЈА АУТОРА

Лични подаци:

Име и презиме: Стеван (Љуба) Кордић
Датум рођења: 20. април 1969. године
Звање: Асистент, Факултет за поморство Котор, Универзитет Црне Горе
Адреса: Доброта бб, 85330 Котор, Црна Гора
Електронска адреса: stevan.kordic@gmail.com

Образовање

Стеван Кордић је уписао Математички факултет у Београду школске 1988/89. године на студијској групи *Математика,* смјер *Рачунарство и математика*. Дипломирао је 13. октобра 1994. године са просјечном оцјеном 8.71 и оцјеном 10 на одбрани дипломског рада и при томе стекао звање дипломираног математичара.

Постдипломске студије је уписао на Математичком факултету у Београду школске 1994/95. Магистарску тезу под насловом *"Једна хеуристика за методу аналитичких таблоа"* одбранио је 19. маја 1998. године. Ментор на постдипломским студијама му је био проф. др Александар Јовановић, ванредни професор Математичког факултета у Београду, а коментори проф. др Жарко Мијајловић и др. Зоран Марковић, директор Математичког института САНУ.

Искуство у настави

На Рударско-геолошком факултету у Београду био је ангажован од 1. јануара 1997. године до 13. априла 1999. Године, гдје је био изабран у звање асистента на предметима Математика I, Математика II и Нумеричка математика.

На Факултету за поморство Котор ангажован је као хонорарни сарадник од 1. јануара 2000. до 31. маја 2000. године, а затим као асистент на математичкој групи предмета од 1. јуна 2000. до данас. На истом факултету је од 5. јула 2006. до 2011. године обављао функцију руководиоца студијског програма Наутика.

У наставном процесу био је ангажован на извођењу предавања и вјежби из сљедећих предмета:

1. Математика I на Одсјеку за рударство, Рударско-геолошког факултета у Београду код проф. др Милоша Миличића
2. Математика на студијском програму Наутика, Факултета за поморство Котор код проф. др Слободана Симића и доц. Др Николе Михаљевића.
3. Математика на студијском програму Бродомашинство, Факултета за поморство Котор код проф. др Слободана Симића и доц. др Николе Михаљевића.
4. Математика I и Математика II на студијском програму Поморске науке, Факултета за поморство Котор код проф. др Ромеа Мештровића.

5. Вјероватноћа и статистика у поморству на студијском програму Менаџмент у поморству, Факултета за поморство Котор код проф. др Ромеа Мештровића.
6. Инжењерска статистика на специјалистичком програму Политехника, Факултета за поморство Котор код доц. др Николе Михаљевића.

Списак научних и стручних радова

Часописи

[1] PREDRAG JANIČIĆ, **STEVAN KORDIĆ**, ***The Geometry Theorems Prover***, Filomat 9:3, **723-732**, 1996.

[2] **STEVAN KORDIĆ**, ***Jedan algoritam za konstrukciju konveksnog omotača skupa tačaka u ravni***, Zbornik Fakulteta za pomorstvo u Kotoru 20, **391-402**, Kotor 2003.

[3] S. M. PEROVICH, D. V. TOSIC, S. I. BAUK, AND **S. KORDIC**, ***On the Exact Analytical Solutions of Certain Lambert Transcendental Equations***, Mathematical Problems in Engineering, vol. 2011, Article ID 685485, 21 pages, 2011. doi:10.1155/2011/685485

[4] **S. KORDIĆ,** T. DAVIDOVIĆ, N. KOVAČ, AND B. DRAGOVIĆ, ***Combinatorial Approach to Exactly Solving Discrete and Hybrid Berth Allocation Problem***, Applied Mathematical Modelling, 2016. doi: 10.1016/j.apm.2016.05.004

[5] **S. KORDIĆ**, ***Application of Sedimentation Algorithm for Solving Max-SAT Problem***, Mathematica Montisnigri, in press, 2016.

Часописи из дигитализације

[1] ЈОШКО КАТЕЛАН, **СТЕВАН КОРДИЋ**, ***Један предлог дигитализације Аустроугарског катастра у општини Котор***, Архивски записи 1-2/2000, **83-92**, Цетиње 2001.

[2] **СТЕВАН КОРДИЋ**, СНЕЖАНА ПЕЈОВИЋ, ***Дигитална заштита архивске и библиотечке грађе из црквених фондова општине Котор***, Архивски записи 1-2/2000, **93-120**, Цетиње 2001.

[3] **STEVAN KORDIĆ**, SNEŽANA PEJOVIĆ, ***Manuskripti u Franjevačkoj biblioteci samostana Sv. Klare u Kotoru***, Godišnjak Pomorskog muzeja u Kotoru L, **265-275**, Kotor 2002.

[4] **STEVAN KORDIĆ**, ***Četiri biblije pisane karolinom iz Franjevačke biblioteke Sv. Klare u Kotoru***, Godišnjak Pomorskog muzeja u Kotoru L, **392-421**, Kotor 2002.

Међународне конференције

[1] PREDRAG JANIČIĆ, **STEVAN KORDIĆ**, **EUCLID** – ***The Geometry Theorems Prover***, Algebra, Logic & Discrete Mathematics, 52, Niš 1995.

[2] **STEVAN KORDIĆ**, ***One Aspect of the Tree Structure Optimization for the Automatic Theorem Proving***, Kurepa's Symposium, **86**, Beograd 1996.

[3] **STEVAN KORDIĆ**, ***One Method of Analytical Tableaux for Infinitary Propositional Calculs*** P_{α}, VIII International Conference Algebra & Logic, Novi Sad 1998.

[4] NATAŠA KOVAČ, BRANISLAV DRAGOVIĆ, **STEVAN KORDIĆ**, (2009), ***Short literature survey of berth allocation problem***, Proceeding of XIX International Conference on Material Handling, Constructions and Logistics, MHCL 2009, University of Belgrade, Faculty of Mechanical Engineering, pp. **339-342**. 978-86-7083-672-3.

[5] SLAVICA M. PEROVIĆ, **STEVAN KORDIĆ**, (2009), ***The special Trans function theory and quadratic equations***, Proceedings of 9th International conference "Research and development in Mechanical industry", RaDMI 2009, Vol. III, Special issue, Marine industry, Kotor, Montenegro, November 2009, Conference Proceedings, pp. **1341-1348.**

[6] **STEVAN KORDIĆ**, NATAŠA KOVAČ, ***One Combinatorial Algorithm for Berth Allocation Problem***, Fourh Workshop on Formal and Automated Theorem Proving and Applications, Belgrade 2011.

[7] STUPALO, V., **KORDIĆ, S.**, ĆOROVIĆ, B., (2011), ***Opportunities for common learning actions within Marco Polo II programme in Croatia and Montenegro***, Proceedings of 6th International Conference on Ports and Waterway - POWA 2011, Zagreb, Croatia, pp. **1-6.**

[8] **STEVAN KORDIĆ**, BRANISAV DRAGOVIĆ, TATJANA DAVIDOVIĆ, NATAŠA KOVAČ, (2012),
A Combinatorial Algorithm for Berth Allocation Problem in Container Port, International Association of Maritime Economists Conference 2012, Taipei, Taiwan

[9] **STEVAN KORDIĆ**, NATAŠA KOVAČ, ŽELJKO PEKIĆ, (2012), ***An Analysis of Estimation and Rearrange Heuristic for Sedimentation Algorithm for Solving Berth Allocation Problem in Container Port***, Book of Proceedings, 4th International Maritime Science Conference, Split, Croatia, pp. **71-79.**

[10] **STEVAN KORDIĆ**, NATAŠA KOVAČ, TATJANA DAVIDOVIĆ, (2015), ***Divide and Conquer Approach to Descrete Berth Allocation Problem***, The 12th Balcan Conference on Operational Research BALCOR 2015, In: Volume: "Mircea cel Batran" Naval Academy Scientific Bulletin, XVIII (2), Constanta, Romania, pp. **307-316.**

Међународне конференције из дигитализације

[1] JELENA ANTOVIĆ, JOŠKO KATELAN, **STEVAN KORDIĆ**, SNEŽANA PEJOVIĆ, ***Prezentacija srednjovjekovnih rukopisnih i štampanih knjiga Istorijskog***

arhiva Kotor na CD-ROM-u, XXI Posvetovanje o strukovnih in tehničnih vprašanjih v arhivi, **198-208**, Radenci 1999.

[2] JELENA ANTOVIĆ, **STEVAN KORDIĆ**, ***Pregled fondova i zbirki Istorijskog arhiva kotor i njegov prikaz na CD-ROM-u*, XXI Posvetovanje o strukovnih in tehničnih vprašanjih v arhivi**, **228-234**, Radenci 2000.

[3] **STEVAN KORDIĆ**, SNEŽANA PEJOVIĆ, ***Manuscripts in the library of thr Franciscan monastery of Santa Clara in Kotor***, Church Archives and Libraries, **323-336**, Kotor 2002.

Домаће конференције

[1] PREDRAG JANIČIĆ, **STEVAN KORDIĆ**, ***EUKLID – Dokazivač geometrijskih teorema***, izlaganje na Jednodnevnom seminaru-konferenciji iz Matematičke logike, Beograd 1993.

[2] PREDRAG JANIČIĆ, **STEVAN KORDIĆ**, ***Jedan pristup aksiomatskom zasnivanju geometrije***, IX Kongres matematičara Jugoslavije, **38**, Petrovac 1995.

[3] **STEVAN KORDIĆ**, ***One Algorithm for Computing the Convex Hull***, XI Yugoslav Geometrical Seminar, Divčibare, 1996.

[4] GORAN OBRADOVIĆ, ALEKSANDAR JOVANOVIĆ, **STEVAN KORDIĆ**, SAŠA MALKOV, ***CCD mikroskopija, astronomija i problemi akvizicije i obrade EEG signala***, XI INFOTEH, **215-217**, Lepenski vir 1996.

[5] JELENA ANTOVIĆ, JOŠKO KATELAN, **STEVAN KORDIĆ**, SNEŽANA PEJOVIĆ, ***CD-ROM Istorijskog arhiva Kotor***, XXIV Komunikacije, **36-37**, Perast 1999.

Пројекти

За вријеме постдипломских студија и након њих, био је ангажован на више научних пројеката и семинара:

[1] Математичка логика и рачунарство, пројекат Математичког института САНУ под руководством проф. др Жарка Мијајловића.

[2] Семинар за теоретску и примјењену логику, пројекат Математичког института САНУ, којим је руководио проф. др Александар Крон, а затим проф. др Ђорђе Вукомановић, на семинару сам био активан од 1993. Године и на њему сам више пута учествовао као предавач.

[3] Примена рачунара у билолошким наукама, пројекат Математичког Факултета у Београду и Биолошког института “Синиша Станковић” из Београда, под руково-дством проф. др Александра Јовановића. У склопу овог пројекта био сам и радно ангажован у Биолошком институту “Синиша Станковић” у Лабораторији за електро-енцефалографију, гдје сам радио на проблемима аквизиције можданих сигнала и њиховој обради и архивирању

[4] Електронско архивирање и презентација културног блага, пројект Математичког института САНУ, под руководством проф. др Жарка

Мијајловића. Резултат овог пројекта су били, између осталих, и реализација неколико CD-ROM-ова:

(i) Статут града Котора (штампана верзија из 1616. године), Прва судско-нотарска књига (1326—1335), избор од преко 200 најзначајнијих докумената архива и преглед културно-историјских споменика са подручја которске општине. Овај је реализован у сарадњи са Историјским архивом Котор.

(ii) Цјелокупна дела Богдана Гавриловића (истакутог математичара, један од оснивача Универзитета у Београду).

(iii) Гравире Београда из колекције Музеја града Београда.

(iv) CD-ROM посвечен 125 година од оснивања Математичког факултета у Београду.

[5] Израда АРХИС информационог система за Државни архив Црне Горе, пројекат математичког факултета у Подгорици. У оквиру овог пројекта био сам ангажован на изради *web site*-а Државног архива Црне Горе.

[6] Уопштавање теорије специјалних тран функција и примјене на нелинеарне система, пројекат Факултета за поморство Котор, под руководством проф. др Славице Перовић. Пројекат је трајао од 2006. до 2007. године.

[7] О егзактности аналитичких рјешења неких фамилија инверзних проблема, пројекат Факултета за поморство Котор, под руководством проф. др Славице Перовић. Пројекат је трајао од 2008. до 2010. године.

[8] *ICT* и *E-Learning* у интермодалном транспорту, билатерални пројекат Министарства образовања и шпорта Републике Хрватске, Министарства науке Црне Горе, Факултета прометних знаности Загреб и Факултета за поморство Котор, под руководством Б. Драговића и З. Кавран. Пројекат је у току (2011—2013).

Прилог 1.

Изјава о ауторству

Потписани-а **Стеван Љ. Кордић**

број уписа

Изјављујем

да је докторска дисертација под насловом

Метод седиментације и његове примјене у проблемима дискретне математике

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

Потпис докторанда

У Београду, 14. јун 2016.

_____

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора: **Стеван Љ. Кордић**

Број уписа:

Студијски програм: **Математика**

Наслов рада: **Метод седиментације и његове примјене у проблемима дискретне математике**

Ментор: **др Жарко Мијајловић, редовни професор**

Потписани **Стеван Љ. Кордић**

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

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

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

Потпис докторанда

У Београду, 14. јун 2016.

_____

Прилог 3.

Изјава о коришћењу

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

Метод седиментације и његове примјене у проблемима дискретне математике

која је моје ауторско дело.

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

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

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

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

Потпис докторанда

У Београду, 14. јун 2016.

_____

1. Ауторство - Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.