Korištenje metode dinamičkog programiranja u rješavanju problema investiranja

Marin, Mačinković

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:205381

Rights / Prava: Attribution-ShareAlike 3.0 Unported/Imenovanje-Dijeli pod istim uvjetima 3.0

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marin Mačinković

KORIŠTENJE METODE DINAMIČKOG PROGRAMIRANJA U RJEŠAVANJU PROBLEMA INVESTIRANJA

DIPLOMSKI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marin Mačinković

JMBAG: 0016116021

Studij: ***Baze podataka i baze znanja***

KORIŠTENJE METODE DINAMIČKOG PROGRAMIRANJA U RJEŠAVANJU PROBLEMA INVESTIRANJA

DIPLOMSKI RAD

Mentorica:

Doc.dr.sc. Nikolina Žajdela Hrustek

Varaždin, svibanj 2021.

Marin Mačinković

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Za potrebe rada, teoretski i praktično analizirana je i primjenjena metoda dinamičkog programiranja za rješavanje problema investiranja. Analiziran je i definiran sam problem investiranja, kao i općeniti pojam investiranja. Također, prikazan je algoritam dinamičkog programiranja sa svojim koracima rješavanja. Detaljno je analizirana i proučena navedena metoda, tako da se jasno razumije kako je došlo do rješenja. Analiziran je i proces višeetapnog odlučivanja te njegova svrha. Nadalje, opisan je primjer problema investiranja koji je i riješen koristeći metodu dinamičkog programiranja. Za odabrani primjer izrađeno je i programsko rješenje. Implementacija i prikaz rada aplikacije su, također, detaljno prikazani tako da je razumljiv pristup rješavanju. Na kraju su izvedeni zaključci i priložena je korištena literatura.

Ključne riječi: investiranje; problem investiranja; proces višeetapnog odlučivanja; dinamičko programiranje; algoritam; programsko rješenje; implementacija; tehnike rješavanja

Sadržaj

Sadržaj iii
1. Uvod 1
2. Operacijska istraživanja 2
3. Modeli, metode i tehnike operacijskih istraživanja 5
3.1. Linearno programiranje 5
3.2. Nelinearno programiranje 6
3.3. Optimizacija na mrežama 7
3.4. Upravljanje zalihama 8
4. Dinamičko programiranje 10
4.1. Karakteristike dinamičkog programiranja 12
4.2. Terminologija u dinamičkom programiranju 13
4.3. Vrste dinamičkog programiranja 14
4.3.1. Deterministično dinamičko programiranje 14
4.3.2. Probabilistično dinamičko programiranje 15
4.4. Višeetapni proces odlučivanja 16
4.4.1. Klasifikacija 16
4.4.2. Matematička definicija 19
4.4.3. Dobivanje rješenja 19
4.5. Načelo optimalnosti 20
4.5.1. Matematička definicija 21
4.6. Stohastički procesi dinamičkog programiranja 22
4.7. Postupak dinamičkog programiranja 23
5. Investiranje 24
5.1. Tipovi investiranja 25
5.1.1. Investiranje u vrijednosne papire 26
5.1.1.1. Dioničke vrijednosnice 27
5.1.1.2. Dugovne vrijednosnice 28
5.1.1.3. Investicijski fondovi 30
5.1.2. Investiranje u nekretnine 32
5.1.2.1. Porast vrijednosti i prihodi 32
5.2. Dvadeset pravila za uspješno investiranje 33
5.3. Primjer investiranja 34
6. Problem investiranja 36
6.1. Definiranje problema 36

6.2. Izgradnja modela 36
6.3. Računski postupak 38
6.4. Grafički prikaz ovisnosti dobiti o investicijama 40
7. Primjer problema investiranja 42
7.1.1. Opis problema 42
7.1.2. Računsko rješenje problema 43
7.1.3. Grafički prikaz rješenja problema 55
8. Programsko rješenje 56
8.1. Prikaz rada aplikacije 57
8.2. Implementacija 62
8.2.1. Klasa Program.cs 62
8.2.2. Pomoćna klasa PomocUNavigaciji.cs 63
8.2.3. Forma Form1 64
8.2.4. Forma Form2 64
8.2.5. Forma Form3 65
8.2.5.1. Konstruktor i varijable 65
8.2.5.2. Metode 66
8.2.5.3. Događaji 72
9. Zaključak 76
Popis literature 78
Popis slika 80
Popis tablica 81

1. Uvod

Zadatak ovog diplomskog rada je teoretski i praktično analizirati i primijeniti metodu dinamičkog programiranja za rješavanje problema investiranja. Da bi se navedeno moglo obaviti, najprije je potrebno postaviti teorijsku pozadinu za ključne pojmove te ih razraditi. Nakon toga je moguće povezati pojedine pojmove i nastaviti s ciljevima i realizacijom praktičnog dijela rada.

U teorijskom dijelu rada najprije su definirana operacijska istraživanja sa pripadajućim metodama. Za ovaj rad je ključno promotriti i definirati dinamičko programiranje. Na samom početku detaljno je objašnjeno što je dinamičko programiranje i zašto je korisno i važno, te gdje je našlo primjenu. Nakon toga prikazani su koraci u algoritmu dinamičkog programiranja, kao i načelo višeetapnog procesa odlučivanja koje je vrlo važno za navedeni algoritam.

Za shvaćanje konteksta rada definiran je i objašnjen sam pojam investiranja, gdje se koristi, koja je svrha investiranja, zajedno sa svim ostalim relevantnim pitanjima vezanim za investiranje. Priložen je i primjer za razumijevanje principa njegove primjene i kako ono funkcionira u realnim sustavima.

Nakon definiranja investiranja, nadalje dolazi razrada problema investiranja. Navedeni problem prikazan je kroz pojašnjenje teorije s par primjera da bi se navedeno lakše razumjelo.

Nakon što je pojašnjen sav teorijski dio i postavljene osnove za razumijevanje ovog rada, slijedi praktični dio. Praktični dio započinje prikazanim primjerom problema investiranja. Problem je najprije definiran, nakon čega je postavljen i prikazan računski postupak rješavanja. Sljedeći korak je prikaz korištenja dinamičkog programiranja na konkretnom primjeru, radi lakšeg razumijevanja algoritma, zajedno s njegovom primjenom i kako bi se uvidjele sve prednosti koje proizlaze iz korištenja.

U narednom poglavlju slijedi definiranje konkretnog problema investiranja koji je zatim riješen koristeći metodu dinamičkog programiranja. Nakon navedenog, dano je pojašnjenje implementacije i kreiranja programskog rješenja za dani problem kroz nekoliko korisnih koraka.

Na kraju su izvedeni zaključci s najvažnijim dijelovima iz kreiranog rada, zajedno s korištenom literaturom koja seže od znanstvenih članaka do knjiga i prikaza primjera.

2. Operacijska istraživanja

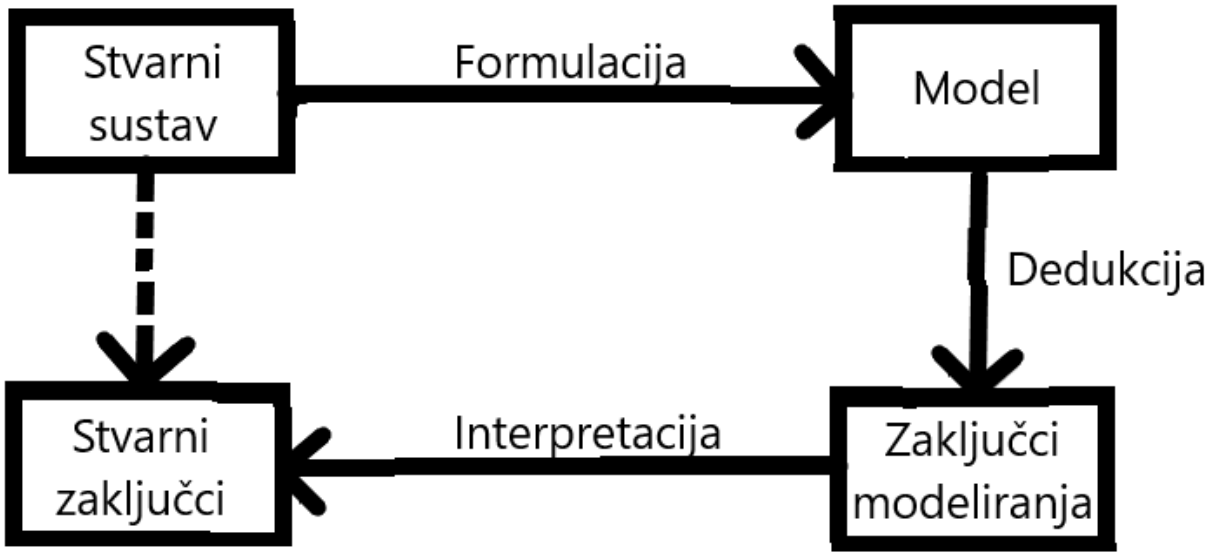
Operacijska istraživanja su interdisciplinarna znanstvena grana u kojoj se primjenjuje metoda dinamičkog programiranja, koje je glavni dio ovog diplomskog rada. Da bi se moglo shvatiti što je dinamičko programiranje, najprije je potrebno definirati i analizirati što su to operacijska istraživanja.

Kroz godine se pojam operacijskih istraživanja mijenjao i usavršavao, te postoji mnogo različitih definicija i objašnjenja istog. Operacijska istraživanja se mogu, s jedne strane promatrati kao skup tehnika i metoda za rješavanje skupa problema. S druge strane, smatra se da je to proces ili aktivnost koja se primjenjuje u svakodnevnom životu. Sam pojam zahtjeva još mnogo istraživanja i ispitivanja da bi se mogao stabilizirati i potpuno precizno analizirati. (Ravindran, A. i suradnici, 2007, str. 3)

Ravindran i suradnici navode sljedeću definiciju u kojoj pokušavaju što bolje i preciznije definirati operacijska istraživanja: „*Operacijska istraživanja podrazumijevaju primjenu metoda znanosti na složene probleme koji se javljaju u usmjeravanju i upravljanju velikim sustavima ljudi, strojeva, materijala i novca u industriji, poslovanju, vladi i vojsci. Prepoznatljiv pristup je razviti znanstveni model sustava koji uključuje mjerenje čimbenika kao što su šansa i rizik, s kojima se mogu predvidjeti i usporediti ishodi alternativnih odluka, strategija ili kontrola. Ukratko, svrha je pomoći menadžmentu da znanstveno utvrdi svoj način djelovanja i upravljanja. Operacijska istraživanja bave se znanstvenom odlukom kako najbolje dizajnirati i upravljati sustavima čovjek-stroj, obično pod uvjetima koji zahtijevaju dodjelu oskudnih resursa*.“ (Ravindran, A. i suradnici, 2007, str. 3-4)

Svrha operacijskih istraživanja leži u konstrukciji i upotrebi modela. Iako se modeliranje mora naučiti iz pojedinačnog eksperimentiranja, postoje načini da se ono teoretski razumije. Model je ovom smislu samo pojednostavljeni prikaz nečega stvarnog. Navedeno sa sobom nosi implikaciju da je model uvijek, nužno, predodžba nečega, i samim time nije savršen i podložan je greškama. Motivacija za upotrebu zamjene umjesto „stvarne stvari“ je ušteda novca i vremena, te stvaranje potrebne prilagodljivosti i pristupa problemu. Ponekad je stvarno okruženje toliko kompliciranо da je reprezentativni model potreban samo da bi ga se razumjelo ili da bi se pomoću njega komuniciralo s drugima. (Ravindran, A. i suradnici, 2007, str. 4)

Na sljedećoj slici prikazan je način izgradnje modela u operacijskim istraživanjima.



Slika 1. Izgradnja modela (Ravindran, A. i suradnici, 2007)

Na slici 1. vidljivo je da: (Ravindran, A. i suradnici, 2007, str. 5-6)

- Na lijevoj strani se nalazi **stvarni sustav** i zaključak stvarnog sustava. Isprekidana crta zapravo predstavlja ono što želimo **zamijeniti** modelom i tako izbjeći velike troškove vremena i novca.
- Izgradnja modela iz stvarnog sustava naziva se **formulacija** i tu se pokušava što je stvarnije moguće preslikati stvarni sustav u njegovu zamjenu, odnosno model.
- **Dedukcijom** se naziva skup tehnika koje ovise o prirodi modela. Podrazumijeva razne formule, programska rješenja i i razne logičke izraze koji su potrebni za rješavanja problema i izvlačenje zaključaka iz istih.
- Zadnji korak je **interpretacija**. Ona zahtjeva opreznu i što precizniju transformaciju zaključala modeliranja u zaključke rada stvarnog sustava. U obzir se moraju uzeti svi mogući nedostaci modeliranja i njihova usporedba sa rješenjem.

Pri izgradnji modela, odnosno pri modeliranju, kako je prikazano na slici 1., pažnju treba obratiti na sljedećih nekoliko vrlo važnih stvari: (Ravindran, A. i suradnici, 2007, str. 7-11)

- Izbjegavati izgradnju prekompliciranog modela ako je moguća jednostavnija konstrukcija,
- Imati na umu oblikovanje problema na način da odgovara tehnici rada,
- Faza dedukcije u procesu izgradnje modela mora biti rigorozno nadzirana,
- Model mora biti validiran od strane stručnjaka prije njegove implementacije,
- Model se nikad ne smije shvaćati preozbiljno jer je podložan mnogim greškama i predstavlja samo prikaz stvarnog sustava,
- Izgradnja modela se ne smije olako shvatiti, niti njegov mogući neuspjeh kritizirati zbog onog za što nikada nije bio namijenjen,
- Izbjeći precjenjivanje modela,
- Koristi modeliranja su usko povezane s izgradnjom modela, pri oba procesa treba biti vrlo pažljiv,
- Model ne može biti precizniji od stvarnog sustava,
- Model ne može zamijeniti stvarni sustav.

U operacijskim istraživanjima koriste se razni modeli, metode i tehnike za dobivanje potrebnih rješenja i pristup modeliranju. Neki od najpoznatijih modela, metoda i tehnika u operacijskim istraživanjima su:

- Dinamičko programiranje,
- Linearno programiranje,
- Nelinearno programiranje,
- Optimizacija na mrežama,
- Upravljanje zalihama.

Metoda dinamičkog programiranja će biti najdetaljnije analizirana budući da je ona temeljni dio ovog rada. Ostale metode će biti ukratko pojašnjene i prikazane da bi se shvatio i širi pristup i rad pri operacijskim istraživanjima. Navedeno će biti prikazano u sljedećem dijelu rada.

3. Modeli, metode i tehnike operacijskih istraživanja

U ovom poglavlju bit će navedeni i ukratko pojašnjeni modeli, metode i tehnike operacijskih istraživanja koji su navedeni u prethodnom dijelu rada, te je za njih rečeno da će biti analizirani. Bit će definirano kako i kada se koriste te za što služe.

3.1. Linearno programiranje

Programski problemi su najčešće usko povezani s raspodjelom ograničenih resursa – materijala, strojeva i novca, na najbolji mogući način tako da se minimiziraju troškovi, a maksimizira prihod. Najbolji način za postizanje navedenog je često postavljanjem i rješavanjem matematičkog problema. Linearno programiranje se bavi navedenim, uz sljedećih nekoliko uvjeta: (Ravindran, A. i suradnici, 2007, str. 13)

- Varijable uključene u rješenje problema ne smiju imati negativnu vrijednost, odnosno moraju biti veće ili jednake nuli.
- Kriterij za određivanje optimalne odluke može se opisati linearnom funkcijom koristeći spomenute varijable.
- Pravila i načela u rješavanju problema mogu se prikazati nizom lineranih jednadžbi ili lineranih nejednakosti. Ovaj prikaz se naziva ograničenje.

Tehnike lineranog programiranja se koriste u raznoraznim područjima za rješavanje kompleksnih problema. Svoju primjenu najčešće nalaze u vojsnim, ekonomskim, industrijskim i društvenim problemima. Tri glavna razloga raznolike upotrebe: (Ravindran, A. i suradnici, 2007, str. 13)

- Veliki broj problema u različitim područjima može se predstaviti ili barem pobliže objasniti modelima linearnog programiranja.
- Linearno programiranje nudi spektar vrlo uspješnih i efikasnih tehnika za rješavanje složenih problema.
- Modeli linearnog programiranja pružaju jednostavnost kojom se može obraditi varijacija podataka.

Za izgradnju modela linearnog programiranja na kojem se temelji rješavanje problema, potrebno je slijediti sljedeća tri koraka: (Ravindran, A. i suradnici, 2007, str. 14)

1. Prepoznati nepoznate varijable koje treba odrediti, te ih predstaviti u smislu algebarskih simbola.
2. Prepoznati sve restrikcije ili ograničenja u problemu i izraziti ih kao linearne jednadžbe ili nejednadžbe koje su linearne funkcije nepoznatih varijabli.
3. Prepoznati cilj ili kriterij, te ga predstaviti kao lineranu funkciju optimalne varijable, koju treba maksimizirati ili minimizirati.

Primjer metode lineranog programiranja je simpleks metoda. To je najznačajnija, ali i najopćenitija metoda lineranog programiranja. U simpleks metodi se primjenjuje iterativni postupak za rješavanje linearnog programa kojim se konstruira neko moguće inicijalno rješenje. Nakon konačnog broja koraka dolazi se do optimalnog rješenja ili se dolazi do zaključka da problem nema optimalno rješenje. (Vukičević, M., 2000, str. 174)

3.2. Nelinearno programiranje

Kao što postoje razni problemi koji se mogu riješiti metodom linearnog programiranja, postoje i mnogi koji se ne mogu – ti problemi se nazivaju problemima nelinearnog programiranja. Metode koje se koriste za rješavanje skupa kompleksnih nelinearnih problema klasificiraju se kao koraci u algoritmu nelineranog programiranja.

Glavni nedostatak u proučavanju područja nelinearnog programiranja je širok spektar tehnika koje se koriste za rješavanje problema. Problem nelinearnog programiranja karakteriziraju pojmovi skupina pojmova koji uključuju nelinearne funkcije. Najbolji primjer navedenog bi bile matematiče funkcije sinus i kosinus. (Ravindran, A. i suradnici, 2007, str. 487)

Kod proučavanja metode linearnog programiranja postoji struktura koju je potrebno slijediti da bi se došlo do rješenja, i na neki način olakšala problematika. Pri rješavanju takvog problema do optimalnog rješenja se dolazi nizom linearnih jednadžbi koje nose zaključak postoji li optimalno rješenje ili ne. Kod nelinearnog programiranja

to nije slučaj, vrlo je teško prepoznati ispravno rješenje te ga interpretirati na ispravan način. Mnogo metoda se pokušalo koristiti za uspješno rješavanje ovih problema, međutim samo mali dio njih se pokazao uspješnima. Uspješnost se još smanjuje kada se radi sa stvarnim problemima gdje mnoge tehnike rada gube korak. (Ravindran, A. i suradnici, 2007, str. 487)

Da bi se osmislio postupak i odabrala prava metoda za rješavanje problema nelinearnog programiranja, potrebno je promotriti sljedeća tri kandidata: (Ravindran, A. i suradnici, 2007, str. 491)

- Sve točke u kojima su prve derivacije jednake nuli.
- Sve točke za koje postoje diskontinuiteti za prvu derivaciju.
- Točke na granicama prostora rješenja.

Pri razvoju osnovnih tehnika i koncepata nelinearnog programiranja, ponajbolji i najosnovniji koncept rada koji se mora svladati je Taylorovo proširenje serije koje podrazumijeva upotrebu i korištenje ploha. Upotreba ove serije ploha u prepoznavanju optimalnih rješenja, karakterizaciji stacionarnih točaka, te u tehnikama linearizacije i kvadratne aproksimacije je jedna od najboljih praksi pri rješavanju nelinearnih problema. (Ravindran, A. i suradnici, 2007, str. 494)

3.3. Optimizacija na mrežama

Mreža je sustav linija i kanala koje povezuju razne točke. Kada govorimo o stvarnom svijetu, pojam mreža je vrlo prisutan, koristi se kod komunikacija, pri spajanju mjesta tračnicama, pri isporuci stvari i putevima isporuka i slično. Problemi se javljaju kada nije poznato hoće li se željene točke uspješno povezati i na koji način. U većini slučajeva za rješavanje mrežnih problema uspješno se koriste već spomenuti algoritam lineranog programiranja, odnosno njegova simpleks metoda. Međutim, za određene probleme postoje efikasnije i lakše metode za postizanje potrebnih rezultata. (Ravindran, A. i suradnici, 2007, str. 73)

Neki od mrežnih problema koji se često pojavljuju u svijetu su: (Ravindran, A. i suradnici, 2007, str. 73)

- Nacionalni lanac trgovina zainteresiran je za isporuku posebnih proizvoda u svoja maloprodajna mjesta iz različitih skladišta. Problem je pronaći optimalni plan otpreme koji bi minimizirao ukupne troškove prijevoza.
- U strojarnici se skupina poslova dodjeljuje grupi strojeva. Problem je utvrditi koji raspored poslova na strojevima maksimizira ukupnu učinkovitost trgovine.
- Ako bi se dodijelili jednosmjerni prometni znakovi za određenu mrežu cesta, problem je pronaći optimalni plan, odnosno shvatiti koja bi odluka smanjila protok prometa na toj cesti i povećala broj korisnika autocesta.
- Tvrtka za prijevoz ima tablicu udaljenosti između gradova, te želi pronaći najkraću rutu i najkraću udaljenost između svih parova gradova. Problem je kako naći najkraću s ciljem dizajniranja učinkovitih puteva za vozače.

Ova skupina problema se još naziva i problemom prijevoza te traži brza i efikasna rješenja. Za rješavanje navedenog često se koristi metoda najkraćeg puta, koja je jedna od najefikasnijih metoda rješavanja navedenog problema. Mrežni dijagram i metoda najkraćeg puta bit će prikazani nešto kasnije u radu, u dijelu u kojem se objašnjava postupak rješavanja problema investiranja pomoću metode dinamičkog programiranja.

3.4. Upravljanje zalihama

Metoda upravljanja zalihama je evoluirala od početnog stadija gdje se računalo samo nekoliko jednostavnih i ključnih parametara do sve detaljnijeg i prikladnijeg rješenja sa mnoštvom korisnih podataka. Ključna stavka u razvoju ovog modela je bilo unaprjeđenje limita. Naime, glavni problem je bio to što je model bio limitiran na rad sa samo jednim proizvodom odjednom, što naravno ne preslikava dovoljno dobro stanje u stvarnom sustavu. Navedeni problem je uspješno riješen upotrebom raznih računalnih programa i izračuna, te je ovaj model sve precizniji i precizniji kako vrijeme odmiče. (Ravindran, A. i suradnici, 2007, str. 344)

Modeli naravno ne nude precizne izračune i formule, međutim dovoljno dobro preslikavaju stanje stvarnog sustava. Jedan od razloga za upravljanje i kontroliranje

zaliha je izbjegavanje gubitka vremena, troškova, smetnji i slično. S druge strane, nadopunjavanje zaliha bi samo rijetko značilo velike zalihe. Očito je da je u pitanju neka vrsta kompromisa, a modeli iz ovog odjeljka čine ovaj kompromis eksplicitnim. To nisu jedina opravdanja za kontrolu zaliha, jer, naravno, postoje i drugi razlozi za držanje zaliha. Zajednička značajka ovih modela je da pretpostavljaju da je potražnja potpuno predvidljiva. U onim situacijama u kojima nije, oni ne predstavljaju pravilnu razmjenu. Modele ne treba kritizirati zbog toga - oni jednostavno nisu namijenjeni predstavljanju tih situacija. Istodobno, treba imat na umu značaj ovog ograničenja. (Ravindran, A. i suradnici, 2007, str. 344)

Problemi upravljanja zalihama najčešće se javljaju kada se smatra da model može potpuno replicirati problem stvarnog sustava, što naravno nije ispravno. Za mnoge probleme koji uključuju tisuće i tisuće zaliha kojima se treba upravljati još uvijek nema ispravnog i potpuno točnog rješenja. Ipak, ovaj model pruža dobru podlogu za kvalitetno odlučivanje na limitiranom broju slučaja i u tim slučajevima je vrlo koristan. (Ravindran, A. i suradnici, 2007, str. 370)

Za navedene probleme i njihovo rješenje se često koristi i algoritam dinamičkog programiranja. Navedeni algoritam će biti prikazan kasnije u radu, pri rješavanju problema investiranja. Na sličan način uz par izmjena u postavci rješenja uspješno se rješava i problem upravljanja zalihama.

4. Dinamičko programiranje

Dinamičko programiranje je vrlo širok pojam i postoje mnoge definicije i interpretacije njegovog značenja.

Winston (1994) za dinamičko programiranje navodi da je to: „*tehnika koja se može koristiti za rješavanje mnogih problema s optimizacijom. U većini aplikacija dinamičko programiranje dobiva rješenja radeći unatrag od kraja problema prema početku, čime se veliki, nezgrapni problem razbija na niz manjih, lakše rješivih problema*.“ (Winston, W. L., 1994, str. 961)

Petrić (1979) ističe važnost planiranja i upravljanja: „*Dinamičko programiranje je posebni matematički aparat, koji omogućuje optimalno planiranje višeetapnih procesa upravljanja. Optimalno upravljanje je ono upravljanje koje daje najbolje rješenje u smislu postizanja maksimalnog cilja operacije*.“ (Petrić, J. J., 1979, str. 213)

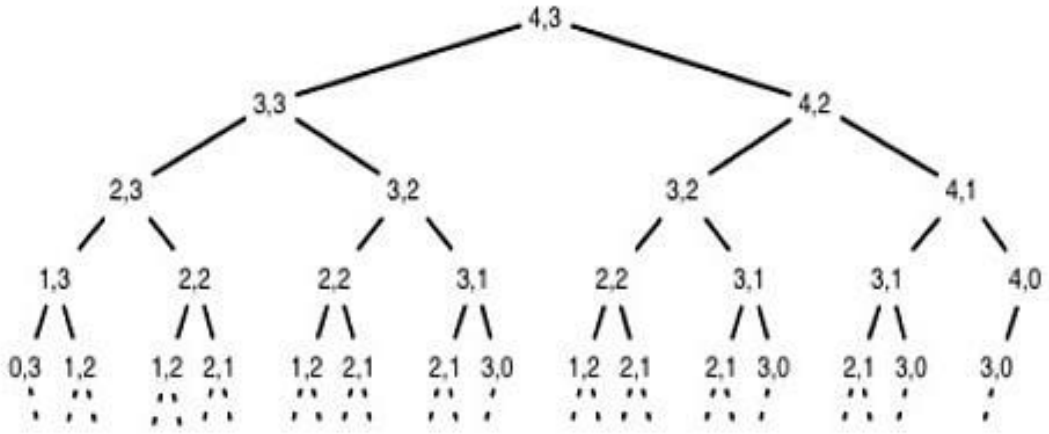
Dobrenić (1978) ističe matematičku stranu: „*Dinamičko programiranje je matematička teorija razrađena za rješavanje problema u kojima je karakterističan proces višeetapnog odlučivanja. Dinamičko programiranje je grana primjenjene matematike i ne smije se miješati s pojmom kompjutorskog programiranja koje predstavlja izradu programa za rješavanje određenih problema na elektroničkom računalu*.“ (Dobrenić, S., 1978, str. 319)

Proces na kojem se zasniva metoda dinamičkog programiranja, odnosno rješavanje po fazama gdje se u proračunu svake faze koriste optimalne vrijednosti dobivene u prethodnoj fazi naziva se Bellmanov princip. Rezultat navedenog principa je slijed optimalnih odluka koje se uspješno primjenjuju kod niza problema. Navedeno ima široku primjenu u gotovo svim segmentima operacijskih istraživanja, od linearnog i cjelobrojnog programiranja, preko analiza mreža do problema skladištenja i slično. Zavisno o inventivnosti i inteligenciji operacijskog istraživača, ovisi hoće li se prepoznati problem koji se može riješiti metodom dinamičkog programiranja i hoće li se smisliti način kako to učiniti. (Kalpić, D., i Mornar, V., 1996, str. 98)

Ova metoda svoju primjenu pronalazi i kod rada sa stablom odlučivanja gdje se različiti korijeni rješavaju posebno. Odnosno, lako se primjenjuje proces postepenog napredovanja po stupnjevima na kojima se uvijek izračunavaju ukupna stanja. Bitna stavka dinamičkog programiranja koje se koristi u stablu odlučivanje je i to što se na

svakom stupnju uspoređuju sva sadržajno identična djelomična rješenja. U tom postupku se odbacuju ona rješenja koja sigurno ne vode boljem rješenju od nekog drugog sadržajno identičnog djelomičnog rješenja. (Barković, D., 2001, str. 232)

Primjer načina na koji radi dinamičko programiranje prikazan je jednostavnim primjerom na sljedećoj slici.



Slika 2. Primjer načina rada dinamičkog programiranja (Podrijetlo izraza, 2020)

Sa slike je očito da je glavna svrha ove metode raspodjeliti problem na više potproblema. Zatim se ti potproblemi razlažu na još sitnije probleme i tako do sitnih detalja. Navedeno je korisno jer se složeniji i kompleksniji problemi mogu jednostavnije riješiti. Također, ukoliko je u pitanju problem na razini određene tvrtke koji se rješava dinamičkim programiranjem, na isti problem se može uključiti više radnika tako da svako uzme jedan dio ukupnog problema. Na kraju se jednostavno usporede dobivena rješenja i izvede se konkretno i optimalno rješenje.

Najistaktnutija stavka je to što je ova metoda zapravo poseban način pristupa rješavanju problema koji nema striktnu formulaciju, odnosno nema striktni niz koraka koji se mora slijediti da bi se došlo do rješenja. To se postiže tako što se izbjegava probleme vezane uz klasične varijante pristupa računanju. U tim varijantama pristupa se računanju odbrojavanjem svih događaja, dok se kod dinamičkog programiranja pretpostavlja postupak istraživanja što je daleko uspješnije i efektivnije. Postupak istraživanja omogućava rješavanje mnogih tipova višeetapnih procesa odlučivanja (Dobrenić, S., 1978, str. 320). O navedenom će detaljnije biti rečeno u nastavku rada.

4.1. Karakteristike dinamičkog programiranja

Iako je u prethodnom poglavlju rečeno da dinamičko programiranje nema striktan niz koraka koje se može slijediti, ipak postoje neki koraci koji omogućavju lakše prepoznavanje uvjeta u kojima je prigodno koristiti ovu metodu, te ju inicijalno lakše prepoznati.

Osnovne značajke koje karakteriziraju metodu dinamičkog programiranja, i koriste se pri njenom definiranju, predstavljene su kroz nekoliko sljedećih koraka: (Andreani, P., 2018, str. 538-541)

1. **Problem** se može podijeliti u **faze**, uz donošenje odluke o politici u svakoj fazi.
2. Svaka faza ima **niz stanja** povezanih s početkom te faze.
3. **Učinak odluke** o politici u svakoj fazi je **transformiranje** trenutnog stanja u stanje povezano s početkom sljedeće faze (prema raspodjeli vjerojatnosti).
4. Postupak rješenja osmišljen je kako bi se pronašla **optimalna politika** za cjelokupni problem, tj. postupak odabira optimalne odluke odnosno odabira optimalne politike u svakoj fazi za svako moguće stanje.
5. S obzirom na trenutno stanje, **optimalna politika** za preostale faze **neovisna** je o odlukama politike donesenim u prethodnim fazama. Stoga optimalna neposredna odluka ovisi samo o trenutnom stanju, a ne i o tome kako se do nje došlo. Ovo je princip optimalnosti za dinamičko programiranje, a o njemu će detaljnije biti govora kasnije u radu.
6. Postupak rješenja započinje pronalaženjem optimalne politike za **posljednju fazu**.
7. Dostupan je **rekurzivni odnos** koji identificira optimalnu politiku za fazu n, s obzirom na optimalnu politiku za fazu n + 1.
8. Kada koristimo rekurzivni odnos, postupak rješenja **započinje na kraju** i pomiče se **unazad** za fazom - svaki put pronalazeći optimalnu politiku za tu fazu. Navedeno se provodi sve dok se ne pronađe optimalna politika s kojom se započinje početna faza. Ova optimalna politika u konačnici daje optimalno rješenje za cijeli problem.

4.2. Terminologija u dinamičkom programiranju

Radi lakšeg shvaćanja narednih potpoglavlja u kojima će biti predstavljeni razni matematički izrazi i jednadžbe, u ovom potpoglavlju će se prikazati sva definirana terminologija koja će se koristiti. Definirana terminologija olakšava korištenje dinamičkog programiranja jer je to zapravo slijed jednostavnih pravila koja se primjenjuju da bi se olakšala formulacija i razumijevanje kompleksnih modela. Također, definirana terminologija se smatra nizom pravila koja se slijede da bi se izbjegle greške, te da bi se lakše interpretirao pojedini model. Navedeno, naravno, smanjuje i vrijeme potrebno za implementiranje pojedine metode i njeno rješavanje.

Terminologija potrebna za razumijevanje daljnjeg izlaganja modela dinamičkog programiranja: (Dobrenić, S., 1978, str. 320-321)

- **Okrugle zagrade – ( )** – za prikazivanje vrijednosti vremena za vremenski zavisne varijable.
    - **Primjer – x (i)** – predstavlja vrijednost vremenski zavisne varijable **x** u vremenu **i.**
- **Uglate zagrade – [ ]** – za prikazivanje argumenta neke od funcionalnih povezanosti.
    - **Primjer – H (i) = H [x (i) – y (i)]** – predstavlja vrijednost **H** u vremenu **i** koja je funkcja razlike **(x – y)** u vremenu **i.**
- **Vitičaste zagrade – { }** – za prikazivanje ekstrema funkcije, odnosno maksimuma ili minimuma funkcije.
    - **Primjer – max { z [x – y] }** – predstavlja maksimalnu vrijednost **z [x – y]** koja se može dobiti podešavanjem **x** i **y**, bez kršenja određenih ograničenja.
- **Brojevi 1, 2, ..., N** – za obilježavanje sukcesivne etape procesa.
    - **Primjer – broj godina** – 1, 2, 3, ..., N – 1, N
- **Broj etapa N, N - 1, ..., 1** – za obilježavanja broja etape od koje se prolazi prema kraju procesa.
    - **Primjer – broj etape od koje se polazi** – N, N – 1, ..., 3, 2, 1

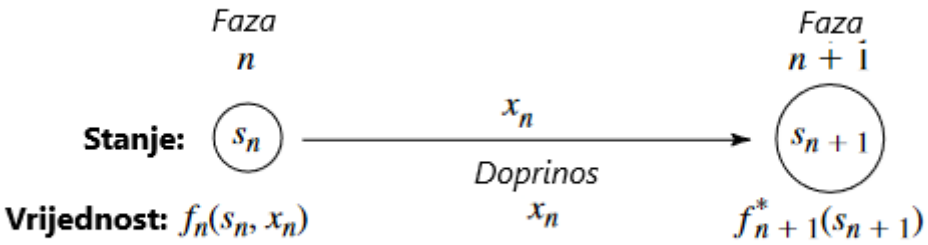
4.3. Vrste dinamičkog programiranja

Vrste dinamičkog programiranja, koje će detaljnije biti obrađene u ovom radu su: (Andreani, P., 2018, str. 541)

- **Deterministično** – stanje u sljedećem stupnju u potpunosti određeno stanjem i optimalnom odlukom u trenutnoj fazi.
- **Probabilističko** – stanje nije u potpunosti određeno stanjem u trenutnoj fazi, postoji vjerojatnost raspodjele za ono što će biti sljedeće stanje.

4.3.1. Deterministično dinamičko programiranje

Kao što je već rečeno, deterministički pristup dinamičkom programiranju se vodi činjenicom da je stanje u sljedećem stupnju u u potpunosti određeno stanjem i odlukom u trenutnoj fazi. Radi lakšeg shvaćanja djelovanja determinističkog programiranja, priložena je sljedeća slika.



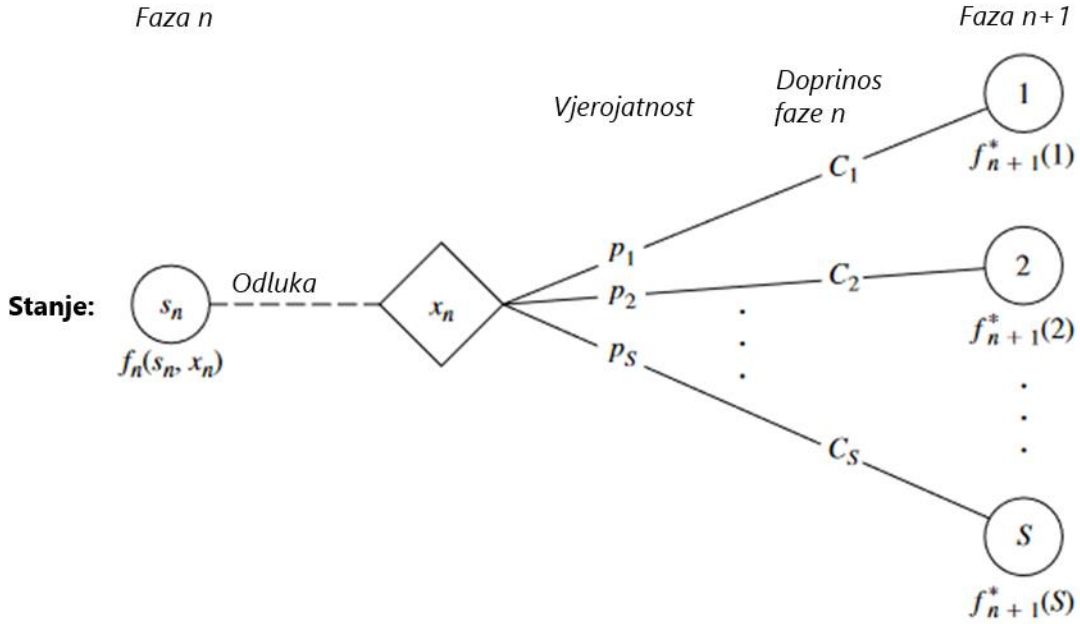
Slika 3. Struktura determinističkog dinamičkog programiranja (Andreani, P., 2018)

Na slici 3. prikazana je struktura determinističkog dinamičkog programiranja. Dakle, u fazi n, proces će biti u nekom stanju s_n. Donošenje odluke x_n tada premješta proces u neko stanje s_{n+1} u fazi $n+1$. Doprinos ciljne funkcije prema optimalnoj odluci prethodno je izračunat sa vrijednošću $f_{n+1}(s_{n+1})$. Odluka x_n također daje određeni doprinos objektivnoj funkciji. Kombinacija ove dvije veličine na prikladan način osigurava $f_n(s_n, x_n)$., doprinos stupnjeva n koji nisu usmjereni na ciljnu funkciju. Optimizacija s obzirom na x_n tada daje $f_n(s_n) = f_n(s_n, x_n)$. Nakon što se pronađu x_n i $f_n(s_n)$ za svaku moguću vrijednost s_n, postupak rješenja spreman

je za pomak unatrag za jednu fazu. Jedan od načina kategorizacije determinističkih problema dinamičkog programiranja je oblik funkcije cilja. Na primjer, cilj može biti minimiziranje zbroja doprinosa iz pojedinih faza, ili maksimiziranje takvog zbroja, ili minimiziranje proizvoda takvih pojmova, i tako dalje. Druga je kategorizacija u smislu prirode skupa stanja za pojedine faze. Konkretno, stanja s_n mogu biti predstavljena diskretnom varijablom ili kontinuiranom varijablom stanja, ili je možda potreban vektor stanja (više od jedne varijable). (Andreani, P., 2018, str. 541-542)

4.3.2. Probabilističko dinamičko programiranje

Kao što je već rečeno, probabilistički pristup dinamičkom programiranju razlikuje se od deterministčkog pristupa dinamičkom programiranja po tome što stanje u sljedećoj fazi nije u potpunosti određeno stanjem i odlukom u trenutnoj fazi. Umjesto toga, postoji raspodjela vjerojatnosti za ono što će biti sljedeće stanje. Međutim, ova raspodjela vjerojatnosti još uvijek je u potpunosti određena stanjem i odlukom u trenutnoj fazi. Radi lakšeg shvaćanja djelovanja probabilističkog programiranja, priložena je sljedeća slika.



Slika 4. Struktura probabilističkog dinamičkog programiranja (Andreani, P., 2018)

Za potrebe ovog prikaza simbolom S označava se broj mogućih stanja u fazi $n+1$ i ta stanja s desne strane označena su kao $1, 2, \dots, S$. Sustav prelazi u stanje i s vjerojatnošću p_i $(\mathrm{i} = 1, 2, \dots, S)$ s obzirom na stanje s_n i odluku x_n u fazi n. Ako sustav prijeđe u stanje i, C_i je doprinos stupnja n ciljnoj funkciji. Kada se slika 4. proširi kako bi obuhvatila sva moguća stanja i odluke u svim fazama, ponekad se naziva stablom odluka. Ako stablo odluka nije preveliko, pruža koristan način sažimanja različitih mogućnosti. Zbog probabilističke strukture, odnos između $f_n(s_n, x_n)$ i $f_{n+1}(s_{n+1})$ nužno je nešto složeniji od onog za deterministično dinamičko programiranje. Točan oblik ovog odnosa ovisit će o obliku sveukupne ciljne funkcije. Da bi se ilustriralo, pretpostavlja se da je cilj minimizirati očekivanu sumu doprinosa iz pojedinih faza. U ovom slučaju, $f_n(s_n, x_n)$ predstavlja najmanji očekivani zbroj faze prema naprijed, s obzirom da su stanje i politika u fazi n, s_n, odnosno x_n. Slijedom toga,

$f_n(s_n, x_n) = \sum_{i-1}^{S} p_i[c_i + f_{n+1}(i)]$, s $f_{n+1}(i) = min_{x_{n+1}} f_{n+1}(i, x_{n+1})$,

gdje se ta minimizacija preuzima preko izvedivih vrijednosti x_{n+1}. (Andreani, P., 2018, str. 562-563)

4.4. Višeetapni proces odlučivanja

U narednom dijelu rada bit će razjašnjeno što su višeetapni procesi odlučivanja te kako se klasificiraju. Također, bit će prikazana matematička definicija procesa, te sam proces dobivanja rješenja.

4.4.1. Klasifikacija

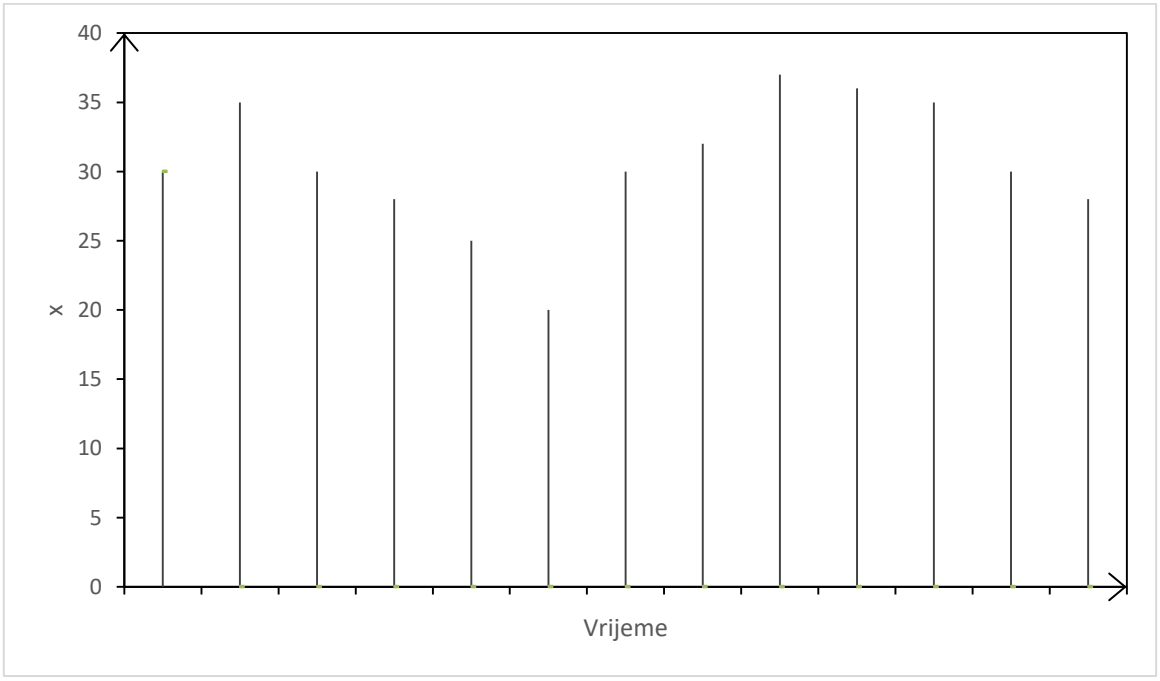
Jacobs (1967) klasificira višeetapne procese odlučivanja na: (Jacobs, O.L.R., 1967., str. 18)

1. **Deterministične** i **probabilistične:**
    - ukazuje na to kako je vrijednost **funkcije** $\boldsymbol{y}$ $[\boldsymbol{x}]$ specificirana vrijednošću njezinog **argumenta x,**
    - vrijednost **deterministične** funkcije je potpuno specificirana vrijednošću njezinog argumenta,

- vrijednost **probabilističke** funckcije ovisi o nepredvidivim, slučajnim varijablama,
- detaljnije objašnjeno u prošlom **potpoglavlju.**

2. Vremenski **diskretne** i vremenski **kontinuirane:**
    - ukazuje na to kako su varijable definirane s obzirom na **vrijeme,**
    - vremenski **diskretna** varijabla je definirana samo skupom diskretnih točaka u prostoru,
        - **Primjer**: serija srednjh dnevnih temperatura u nekom određenom mjestu.
    - **kontinuirani** zapis temperature je vremenski **kontinuirana** varijabla,
    - procesi u kojima su sve varijable diskretne nazivaju se **diskretni procesi,**
    - procesi u kojima su varijable vremenski kontinuirane nazivaju se **kontinuirani procesi,**
    - **slike 5.** i **6.** prikazuju razlike između diskretne i kontinuirane varijable, da bi sve navedeno bilo lakše razumljivo.

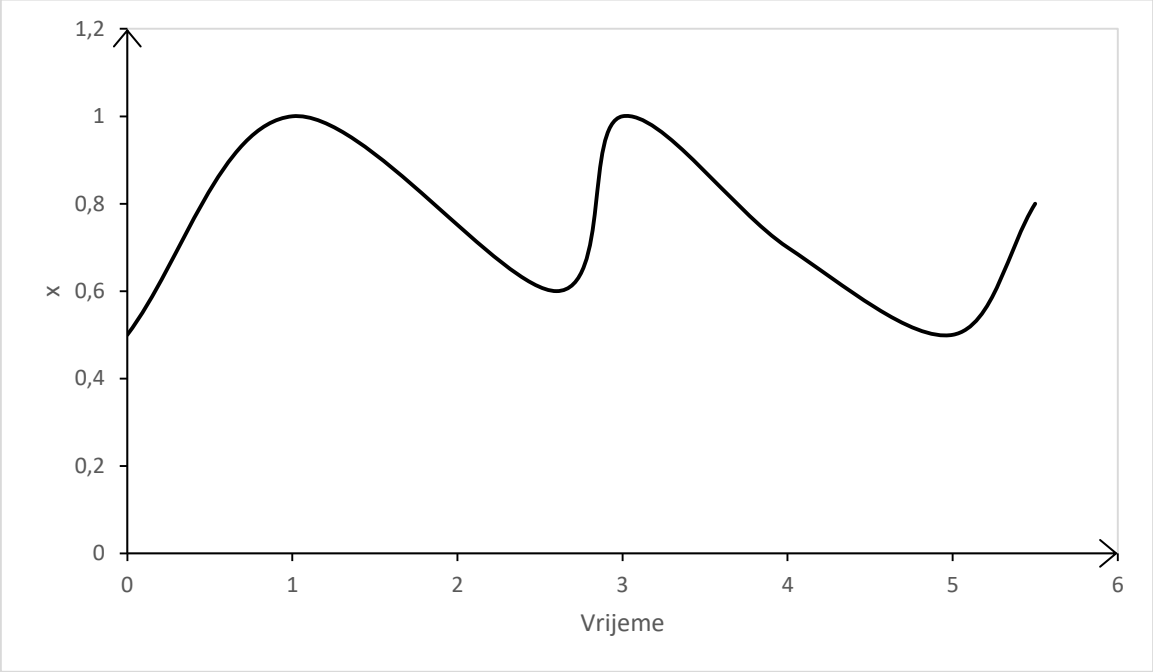
(Dobrenić, S., 1978, str. 323)



Slika 5. Vremenski diskretna varijabla (Dobrenić, S., 1978, str. 324)

Na slici je vidljivo da se vremenski diskretna varijabla mijenja s obzirom na skup diskretnih točaka u prostoru. Odnosno, vidljivo je kako se ta varijabla mijenja u vremenu.

Za razliku od diskretne varijable, kako je već rečeno, kontinuirana je predstavjena kao kontinuirani zapis u nekom određenom vremenu. Navedeno je vidljivo na sljedećoj slici.



Slika 6. Vremenski kontinuirana varijabla (Dobrenić, S., 1978, str. 324)

Formuliranje modela za vremenski kontinuirane i probabilističke višeetapne procese, koji ovise o slučajnim varijablama, te njihovo rješavanje zahtjeva detaljnu matematičku analizu i istraživanje. S druge strane, kod vremenski diskretnih varijabli i determinističkih višeetapnih procesa, rješavanje je nešto jednostavnije i lakše za shvaćanje. (Dobrenić, S., 1978, str. 325)

Formuliranje modela i primjena vremenski diskretnih varijabli, te determinističkih višeetapnih procesa koristit će se i dalje kroz rad budući da je navedeno temelj za rješavanje problema investiranja, koje jej ujedno i glavna tema ovog diplomskog rada.

4.4.2.Matematička definicija

Da bi se mogli definirati višeetapni procesi odlučivanja, najprije je potrebno matematički definirati pojam procesa. Proces je označen sa p, te predstavlja točku koja pripada M-dimenzionalnom prostoru R. p_0 označava početno stanje sustava, dok su $p_0, p_1, \dots, p_n$ stanja sustava u uzastopnim vremenskim trenucima, a W predstavlja operator. Ako postoji relacija: (Petrić, J. J., 1979, str. 214)

$$p_0 = p, \;\; p_{n+1} = W(p_n), \;\; n = 0,1,2, \dots$$

koja predstavlja skup vektora $(p_0, p_1, p_2, \dots)$, kao reprezentaciju ponašanja sustava u diskretnim trenucima vremena $n = 0,1,2, \dots$, tada skup vektora definira proces, odnosno posebnu vrstu procesa koja se naziva višeetapni proces. Ova vrsta procesa često se javlja u normalnim uvjetima, kao što su uglavnom procesi u ekonomiji, vojsci i tehnici. (Petrić, J. J., 1979, str. 214-215)

Relaciju jednostavnije možemo prikazati s $p_n = W^n(p)$ što označava da je određeni broj puta, označen sa n, primjenjen operator W. Dakle, može se reći da su višeetapni procesi zapravo definirani početnim stanjem procesa p, i transformacijom operatora $W(p)$, odnosno $[\, p, W(p) \,]$. Ovaj proces možemo promatrati, i najlakše definirati kao ponašanje promatranog sustava u vremenu. (Petrić, J. J., 1979, str. 215)

4.4.3. Dobivanje rješenja

Da bi se dinamičkim programiranjem mogao prikazati višeetapni proces dobivanja rješenja potrebno je definirati i razraditi i pojam rješenja. Već definirani pojam višeetapnog procesa (u prošlom dijelu) možemo proširiti na način da se u transformaciju parametra W uvede još jedan vremensko ovisno parametar. Izborom ovog parametra utjecat će se na proces. Neka je taj parametar q_i u trenutku i, tada je proces definiran nizom vektora: (Petrić, J. J., 1979, str. 219)

$$p_{n+1} = W(p_n, q_n), \qquad n = 0,1, \dots$$

Izbor veličine q_i naziva se određivanje dopustivog rješenja. Navedena veličina se koristi zbog postizanja određenog cilja. Najčešći oblik takve funkcije je $F(p, p_1, p_2, \dots, q_0, q_1, \dots)$, te se ona naziva funkcijom cilja. (Petrić, J. J., 1979, str. 219)

Nakon svega definiranog u prošlom odlomku, konačno možemo definirati i sam višeetapni proces dobivanja rješenja. N-etapni proces je opisan nizom vektora

$[\, p, p_1, p_2, \dots, p_{n+1}, q_0, q_1, \dots, q_N \,]$, gdje je

$$p_{n+1} = W(p_n, q_n) \text{ za } 0 \leq n \leq N$$

Skup dopustivih rješenja takvih da je $q_n = q_n(p, p_1, \dots, p_n, q_0, q_1, \dots, q_{n-1}$ naziva se strategijom. Strategija koja maksimizira funkcije cilja F naziva se optimalnom strategijom. Jedan od temeljnih zadataka dinamičkog programiranja je određivanje upravo optimalne strategije koju je potrebno slijediti. (Petrić, J. J., 1979, str. 219)

4.5. Načelo optimalnosti

Kao što je u prethodnom poglavlju zaključeno, računski postupak dinamičkog programiranja temelji se na pronalaženju optimalne strategije, odnosno temelji se na načelu optimalnosti.

Četirkin (1971) za optimalnost navodi: „*Optimalna politika ima svojstvo, da ma kakvo bilo početno stanje o i optimalna prva odluka u (1), daljnje odluke moraju činiti optimalnu politiku s obzirom na stanje koje rezultira iz prve odluke*.“ (Četirkin, E.M., 1971)

Petrić (1979) navodi sličnu definiciju optimalnosti: „*Za procese dobivanja rješenja, u kojima strategija zavisi samo od tekućeg stanja, optimalna strategija ima takvu osobinu, da bez obzira kakvo je početno stanje sustava i prvobitno rješenje, sljedeće rješenje treba odrediti optimalnu strategiju u odnosu na stanje dobiveno kao rezultat prvobitnog rješenja.*“ (Petrić, J.J., 1979, str 220)

Načelo optimalnosti je ključ uspješnosti dinamičkog programiranja i jedna od njegovih najvažnijih karakteristika i stavki. Ako promatramo neki izabrani $x\,(N)$, nije potrebno ponovno ispitivati sve odluke vezane za pojedinačni izbor $x\,(N)$. Naprotiv, trebaju se ispitivati samo one odluke koje su optimalne za $(N-1)$ etapu postupka s pripadajućim resursima $Q - x\,(N)$. Imajući navedeno na umu, jasno je da zahtjevana računanja nisu multiplikativna, već aditivna, i to onako kako raste broj etapa u procesu. (Dobrenić, S., 1978, str. 322)

4.5.1. Matematička definicija

Proces optimalnosti je radi lakšeg shvaćanja ilustriran na sljedećem primjeru N-etapnog procesa dobivanja rješenja, o kojem je već bilo govora prije u radu. Navedeni proces ima potrebnu osobinu razdvajanja sadašnjeg od prošlog stanja i obrnuto. Neka proces počinje od stanja p, i neka je optimalna strategija zadana s $q_0, q_1, \dots, q_{n-1}$ tako da vrijedi poznata relacija $p_k = W\ (p_{k-1}, q_{k-1})$. Tada skup rješenja $q_1, q_2, \dots q_{N-1}$ mora biti optimalan za $(N-1)$ – etapni proces, koji počinje od stanja p_1. (Petrić, J.J., 1979, str 220)

Matematička definicija načela optimalnosti može se prikazati i na primjeru maksimizacije funkcije cilja, o kojem je također već bilo govora u prethodnim poglavljima rada. Funkcija glasi:

$$F\ (p, p_1, \dots, p_N, q_0, q_1, \dots, q_N) = \sum_{k=0}^{N} G(p_k, q_k)$$

Maksimum funkcije F, po skupu rješenja $q_1, q_2, \dots q_N$, zavisi samo od početnog stanja p i broja koraka N.Označava se s $f_N(p)$.

$$f_N(p) = \underset{[q_1, q_2, \dots, q_n]}{Max}\ F(p, p_1, p_2, \dots, p_N, q_0, q_1, \dots, q_N$$

Primjenjujući princip optimalnosti može se zaključiti da pri bilo kakvom početnom rješenju q_0, vrijedi za $N \geq 1$:

$$G(p, q_0) + [\ G(p_1, q_1) + \cdots, +G(p_N, q_N)\ ] = G(p, q_0) + f_{N-1}, (p_1)$$

Maksimum funkcije dobije se tako da primjenimo gornji izraz, pa slijedi:

$$f_N(p) = \max_{q_0}\big[\ G(p, q_0) + f_{N-1}\big(W(p, q_0)\big)\ \big], N \geq 1, \text{ gdje je}$$

$$f_0(p) = \max_{q_0} G(p, q_0)$$

(Petrić, J.J., 1979, str 221)

4.6. Stohastički procesi dinamičkog programiranja

Stohastiči procesi dinamičkog programiranja se proučavaju na isti način kao i deterministički procesi, odnosno svode se na funkcionalne jednadžbe. Ovakve vrste procesa su vrlo česte i imaju mnogobrojne primjene u stvarnom svijetu, primjerice u vojsci. Kod ovakve vrste procesa stanje sustava nije u potpunosti određeno, nego ovisi o još nekim slučajnim faktorima: (Petrić, J.J., 1979, str 222)

$$p_0 = p$$

$$p_{k+1} = W(p_k, q_k, r_k)$$

gdje r_k predstavlja slučajni faktor.

Nadalje, slučajni faktor se naravno mora uzeti u obzir pri korištenju u svim prametrima i funkciji višeetapnog procesa. Dakle, na proces se u određenoj mjeri može utjecati upravljanjem q_k, na način da se usmjeri u željenom pravcu. Ipak, taj postupak se ne može u potpunosti kontrolirati, jer je ovisan upravo o slučajnom faktoru. Navedeni proces se naziva stohastičkim procesom upravljanja i direktno je vezan za funkciju cilja koju treba ekstremizirati, odnosno:

$$F = \sum_{k=0}^{N} G(p_k, q_k)$$

I za samu funkciju F se može reći da je slučajna jer direktno ovisi o slučajnom faktoru. Za bilo koju strategiju funkcija cilja će ostati slučajna jer će uvijek ovisiti o slučajnom faktoru. Prema tome, nije moguće maksimizirati funkciju cilja, ali se može tražiti takvo upravljanje koje će maksimizirati srednju vrijednost, odnosno predstaviti matematičko očekivanje. To postižemo na način da pretpostavimo da su r_k nezavisne slučajne varijable promjenjive sa istom funkcijom raspodjele vjerojatnosti $dP(r_k)$. Označimo sa $f_N(p)$ maksimum matematičkog očekivanja slučajne funkcije. Koristeći princip optimalnosti dolazi se do rekurzivne relacije: (Petrić, J.J., 1979, str 223)

$$f_N(p) = \max_{q_0}\left[G(p, q_0) + \int f_{N-1}(W(p, q_0, r_0) dP(r_0)\right], f_0(p) = \max_{q_0} G(p, q_0)\Big)$$

Dakle, možemo zaključiti da se koriste isti principi rada kao i kod determinističkog dinamičkog programiranja, te se primjenjuju funkcionalne jednadžbe.

4.7. Postupak dinamičkog programiranja

Na nekin način za rezimiranje samog dinamičkog programiranja, postupak ove metode se može svesti na sljedeća tri koraka: (Dobrenić, S., 1978, str. 322)

- **Definiranje** funkcije optimalnog prihoda odnosno troškova,
- **Izvođenje funkcionalne jednadžbe** za funkciju optimalnog prihoda odnosno trošova. Često puta je prikladno da se ova jednadžba izvede iz načela optimalnosti,
- **Korištenje funkcionalne jednadžbe** da bi se pronašle funkcije optimalne odluke koje daju optimalnu politiku.

Navedeni postupak će se koristiti u daljnjem dijelu rada kada će se raditi sa problemom investiranja te će se slijediti navedeni koraci.

5. Investiranje

U nastavku rada navode se definicije pojma investiranja koje su preuzete od nekolicine autora odnosno nekoliko izvora relevantne literature.

Richardson (1996) za investiranje navodi: *„Općenito se kaže da je ulaganje (investiranje) konzervativno raspolaganje sredstvima u cilju postizanja dugoročnog prihoda. S druge strane špekulacija je kratkoročno kupovanje ili prodaja u nadi da se profitira uslijed promjena na tržištu.“* (Richardson, L., 1996, str. 28)

Bernstein i Damodaran (1998) ističu važnost upornosti u procesu investiranja: *„Ulaganje nije jednostavno i lako napredovanje do bogatstva. To je naporan put koji zahtjeva ustrajnost i vjeru u proces. Dugoročne stope povrata mogu se uvelike promijeniti uklanjanjem malog dijela ukupnog vremena u kojem je ulagač izložen tržištu.“* (Bernstein, P. L., i Damodaran, A., 1998, str. 5)

Orsag (2006) ističe potrebu temeljite analize za uspješne rezultate: *„Investicija je operacija koja, nakon temeljite analize, obećava sigurnu glavnicu i primjeren prinos. Operacije koje ne zadovoljavaju te zahtjeve su špekulativne“* (Orsag, S., 2006, str. 27)

Bernstein i Damodaran (1998) za proces investiranja ističu i sljedeće važne točke: (Bernstein, P. L., i Damodaran, A., 1998, str. 1):

- Opisuje korake u **stvaranju portfelja** i naglašava slijed uključenih radnji, od razumijevanja sklonosti **riziku** investitora do odabira i raspodjele imovine i procjene njihove izvedbe.
- Pruža **strukturu** koja omogućava investitorima da vide izvore različitih investicijskih **strategija** i **filozofija** opisanih u tisku i u investicijskim biltenima te da ih prate do njihovih zajedničkih korijena.
- Naglašava razne komponente potrebne za **uspješnost** investicijske **strategije**, otkrivajući pritom zašto neke strategije koje na papiru izgledaju dobro nikad ne rade za one koji ih koriste.

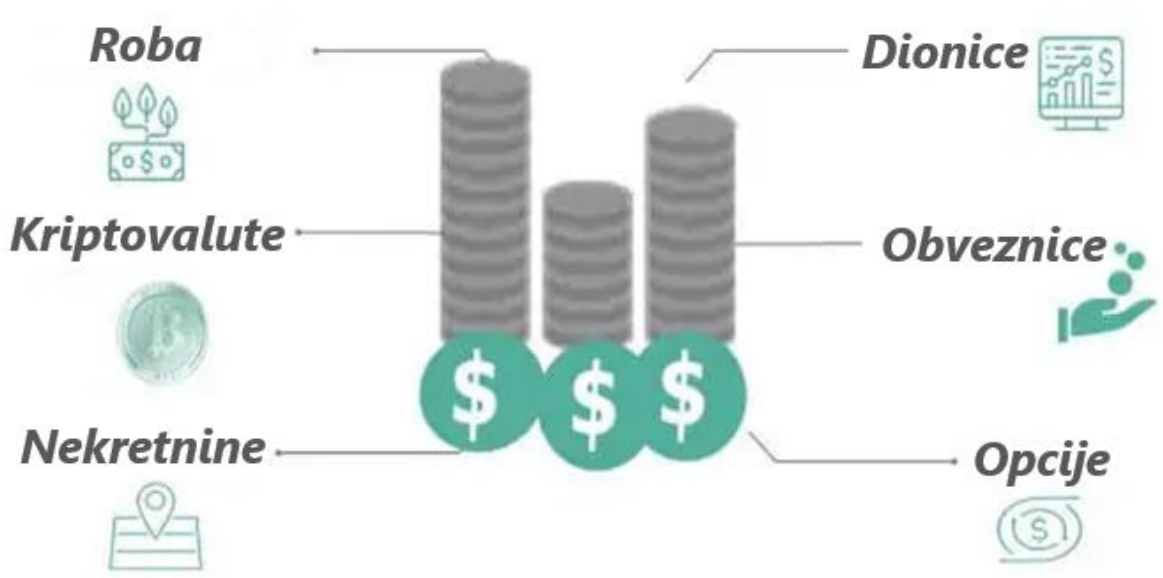
Investiranje se koristi sa svrhom stvaranja određenog prihoda, odnosno za određene ulazne vrijednosti žele se dobiti uvećane, izlazne vrijednosti. Ipak, pažnju treba obratiti na dvije strane investiranja, rizik i povrat. Nizak rizik općenito znači da je i očekivani povrat nizak. Što je povrat veći, to je naravno i rizik veći. Naravno, temeljna

pretpostavka s kojom se ulazi u investiranje je da će se ostvariti određeni prihod. Investiranje s niskim rizikom se najčešće odnosi na obveznice i dionice, dok se veći rizik preuzima kod ulaganja u robe i derivate. (Picardo, E., 2021)

Ulagači mogu samostalno ulagati bez pomoći investicijskog stručnjaka ili zatražiti usluge licenciranog i registriranog investicijskog savjetnika. Navedeno se savjetuje početnicima da se uvedu u svijet investiranja na najbolji način. Tehnologija je također omogućila investitorima mogućnost primanja automatiziranih investicijskih rješenja putem robo-savjetnika koji predlažu najbolju moguću investiciju u odabranom trenutku. Iznos naknade ili novca potreban za ulaganje uvelike ovisi o vrsti ulaganja i ulagačevom financijskom položaju, potrebama i ciljevima. (Picardo, E., 2021)

5.1. Tipovi investiranja

Važan čimbenik investiranja je znati u što treba ulagati. Većina ulagača najprije pomno promotri tržište i sve njegove faktore u danom trenutku nakon čega se odlučuje u kojem smjeru ići. Navedeni proces često zna trajati duže vremena dok se određeni paramteri ne poslože točno onako kako ulagač želi. Može se reći da je investiranje zapravo dugotrajan proces u kojem nema mjesta za brzanje i donošenje nepromišljenih odluka. Sljedeća slika najbolje prikazuje koji su najčešći tipovi investiranja, odnosno u što se najviše ulaže u današnje vrijeme. (Thakur, M., 2021)



Slika 7. Tipovi investiranja (Thakur, M., 2021.)

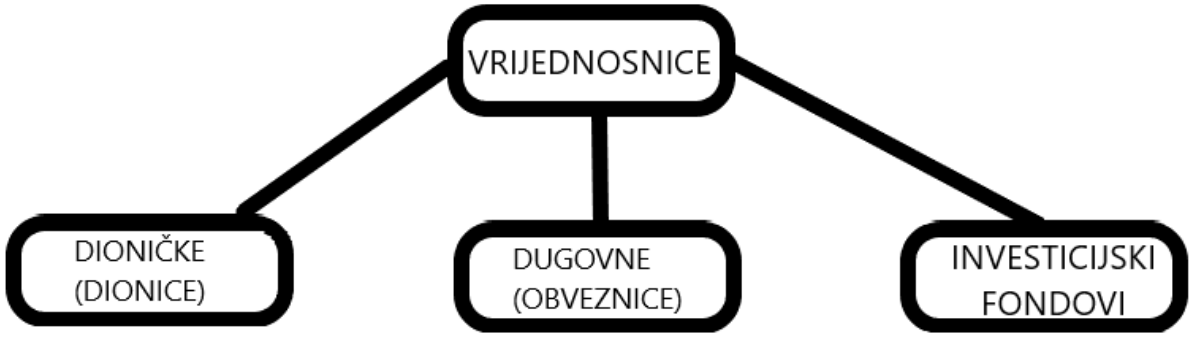
Sa slike je očito da se najčešće investira u: (Thakur, M., 2021.)

- **Dionice** - Tvrtke prodaju dionice i zauzvrat dobivaju novac. Prodaja dionica znači prodaju vlasništva nad tvrtkom u toj mjeri.
- **Obveznice** - Obveznice su instrumenti s fiksnim dohotkom koje društvo izdaje u zamjenu za gotovinu, a takav izdavatelj obveznica duguje imateljima obveznica dug.
- **Opcije** - Ugovor o opcijama je dogovor između dviju strana u kojem jedna strana pristaje kupiti ili prodati određenu imovinu na kasnije dogovoreni datum.
- **Nekretnine** - Pod nekretninama se podrazumijeva imovina, zemljišta, zgrade i slično. Glavna korist od ulaganja u nekretnine bila bi ta da bi došlo do stvaranja bogatstva procjenom vrijednosti nekretnine.
- **Kriptovalute** - Kriptovaluta je digitalna valuta koja ima snažnu kriptografiju za osiguravanje financijskih transakcija i koristi se za provjeru i regulaciju prijenosa sredstava, generiranja novčanih jedinica i slično.
- **Robu** - Primjeri ulaganja u robe uključuju plemenite metalne poluge poput zlata, srebra, platine. Također, prisutni su energetski izvori poput sirove nafte i plina, ili prirodni resursi poput poljoprivrednih i drvnih proizvoda.

5.1.1. Investiranje u vrijednosne papire

Dionice su najznačajniji vrijednosni papiri (vrijednosnice), te se najčešće spominju u burzovnim izvješćima. U modernim tržišnim gospodarstvima praktički nema financijske ili burzovne transakcije koja na ovaj ili onaj način ne bi utjecala na cijenu dionica. Za vrijednosnice se, općenito, može reći da su to svi dokumenti koji predstavljaju pravnu notu, te nose neku vrijednost. Vrijednosnice su različite za svaku državu, te ovise o tome kako ih pojedina država vrednuje. Osnovna podjela vrijednosnica je na dioničke (instrumente dioničkog kapitala) i dugovne. (Cvjetičanin, M., 2004, str 77)

Podjela vrijednosnica vidljiva je i na sljedećoj slici.

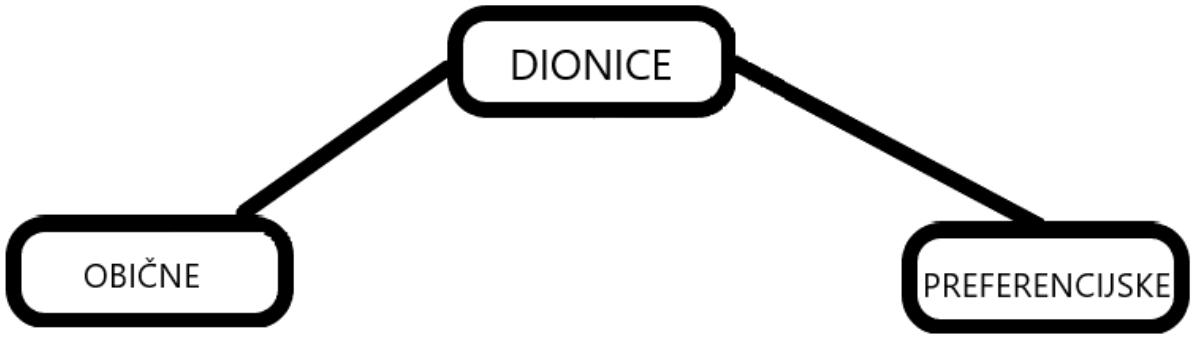


Slika 8. Podjela vrijednosnica (Cvjetičanin, M. , 2005, str. 77)

5.1.1.1. Dioničke vrijednosnice

Kao što je već rečeno, dionice su najznačajnije vrijednosnice, ali ne i jedine. Najznačajnije vrijednosnice se mogu podijeliti u dvije skupine, obične i preferencijske, tako da se lakše mogu promatrati. (Cvjetičanin, M., 2004, str 78)

Podjela dionica vidljiva je i na sljedećoj slici.



Slika 9. Podjela dionica (Cvjetičanin, M. , 2005, str. 78)

Obične dionice su one vrijednosnice koje predstavljaju potvrdu o vlasništvu, ili suvlasništvu određene tvrtke, poduzeća i slično. Osoba ili ustanova koja je vlasnik obične dionice je proporcionalni vlasnik dijela korporacije. Obične dionice svojim vlasnicima donose dva osnovna prava: (Cvjetičanin, M., 2004, str 79)

- pravo da prime dividendu po svakoj dionici kojoj su vlasnici,
- pravo da sudjeluju u glasovanju na skupštini dioničara.

Ipak, glavni razlog kupnje dionica je očekivanje vlasnika dionica da će njihova cijena porasti na tržištu iznad one cijene za koju su ju kupili.

S druge strane, preferencijske dionice su one vrijednosnice koje, isto kao i obične, predstavljaju potvrdu o vlasništvu ili suvlasništvu određene tvrtke, poduzeća i slično. Ipak, preferncijske dionice imaju dvije značajne privilegije u odnosu na obične: (Cvjetičanin, M., 2004, str 80):

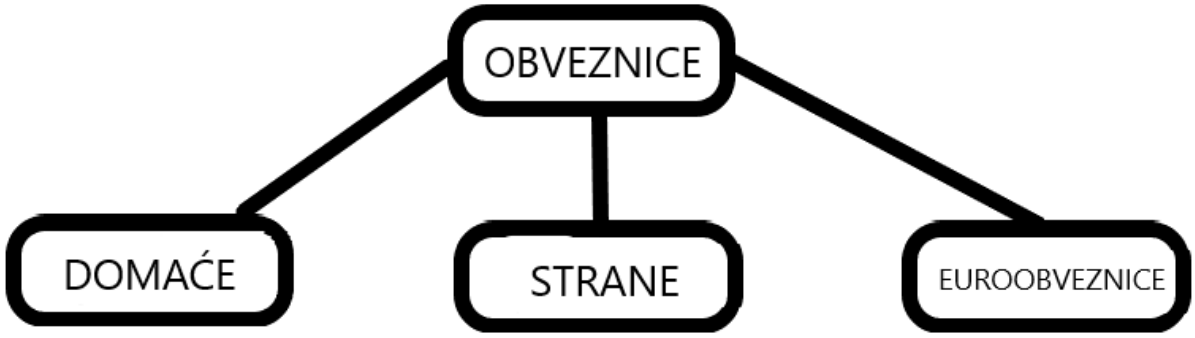
- Vlasnici preferencijskih dionica imaju pravo na dobivanje dividendi prije vlasnika običnih dionica.
- Ukoliko dođe do stečaja, i/ili zatvaranja poduzeća, vlasnici preferencijskih dionica imaju pravo svoja potraživanja iz stečajne mase namiriti prije vlasnika običnih dionica.

Treba naglasiti da ovakva pravna karakteristika preferencijskih dionica uvjetuje činjenicu da kod njih nominalna cijena (osobito kod zatvaranja poduzeća) može imati veliko značenje, dok je kod običnih dionica potpuno nebitna. (Cvjetičanin, M., 2004, str 80)

5.1.1.2. Dugovne vrijednosnice

Obveznice su, uz dionice, najpoznatiji i najpopularniji tip vrijednosnica. Razlikuju se od dionica jer su daleko konzervativnije, odnosno, njihove cijene nemaju dramatične fluktuacije kao cijene dionica. Obveznica je zapravo tip vrijednosnice kojom se izdavatelj obveznice obvezuje vratiti primljenu sumu novca u određenom roku dospijeća, dok se u međuvremenu vlasniku obveznice isplaćuje točna određena kamata, navedena na samoj obveznici. Dakle, može se zaključiti da je ova vrsta vrijednosnica zapravo dugovni instrument putem kojeg se izvodi zajmovni odnos između zajmodavca na jednoj strani i zajmoprimca, odnosno zajmotražioca na drugoj strani. Upravo radi navedenog, obveznice se smatraju najpoznatijim instrumentom stvaranja dugova. (Cvjetičanin, M., 2004, str 97)

Radi lakšeg promatranja obveznica, one se dijele po mjestu i tipu izdavatelja. Na slici 10. vidljiva je podjela po mjestu izdavatelja.



Slika 10. Podjela obveznica po mjestu izdavatelja (Cvjetičanin, M. , 2005, str. 99)

Sa slike 10. je očito da se obveznice po mjestu izdavatelja dijele na: (Cvjetičanin, M. , 2005, str. 99)

- **Domaće** – izdaju ih ustanove (država ili korporacije) koje svoje pravno sjedište imaju u određenoj državi, npr. Republici Hrvatskoj.
- **Strane** – izdaju ih ustanove (strane države ili korporacije) koje svoje pravno sjedište imaju izvan određene države, osim ako ih putem svojih mješovitih ili drugih poduzeća ne izdaju kao domaće obveznice.
- **Euroobveznice** – izdaju ih specifična tržišta koja nisu nigdje stvarno locirana i ne nalaze se pod ničijom jurisdikcijom, već ga stvaraju američke, japanske i europske banke međusobno na europskom kontinentu. Zapravo, to su obveznice koje su za neku zemlju izdane u devizama.

Osim podjele po mjestu izdavatelja, obveznice se dijele i po tipu izdavatelja, kako je već rečeno. Navedena podjela je vidljiva na slici 11.



Slika 11. Podjela obveznica po tipu izdavatelja (Cvjetičanin, M. , 2005, str. 100)

Sa slike 11. je očito da se obveznice po tipu izdavatelja dijele na: (Cvjetičanin, M. , 2005, str. 100-102):

- **Državne** – izdaju ih vlada i drugi državni organi i agencije. Za izvršenje obveze prema vlasnicima obveznica vlada jamči svojim proračunom i pravnom mogućnošću da do sredstava dođe putem oporezivanja svojih subjekata (građana i poduzeća).
- **Korporacijske** – izdaju ih različita poduzeća (trgovačka društva), te predstavljaju dugoročne dugovne obveze poduzeća-izdavatelja, često osigurane dijelom ili cjelokupnom njihovom imovinom.
- **Municipalne** – izdaju ih općine, županije i drugi oblici lokalnih samouprava. Budući da su i to zapravo državni organi, u mnogo slučajeva se ove obveznice svrstavaju pod državne.

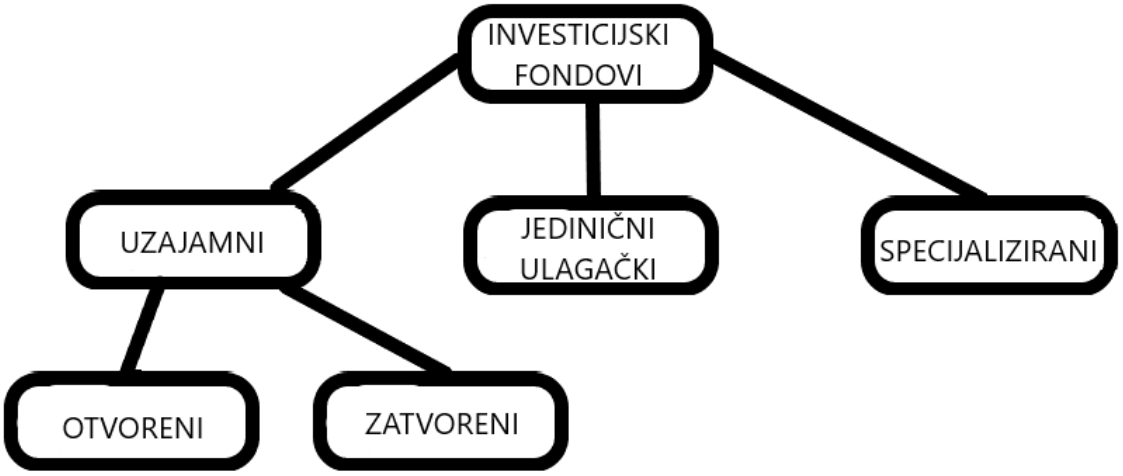
Za razliku od kupnje dionica, kupnjom jedne od obveznica, stječe se kreditni odnos (ne vlasnički). Vlasnik obveznice ima veliku privilegiju u odnosu na dioničare, u slučaju stečaja poduzeća, iz stečajne mase se prvo namiruju vlasnici obveznica, a tek nakon što se oni namire u cijelosti, iz ostatka stečajne mase namiruju se dioničari (prvo preferencijski, a nakon toga obični). Bitna razlika je i to što obveznice svojim vlasnicima donose kamatu (ne dividendu). Kamata je stalna i obećana kod izdavanja obveznice te osigurava primitak prihoda jednom godišnje. (Cvjetičanin, M. , 2005, str. 103)

5.1.1.3. Investicijski fondovi

Investicijski fondovi su organizacije, udruženja ili specifične tvrtke osnovane isključivo sa svrhom prikupljanja sredstava od svojih članova (investitora) i ulaganja tih sredstava u različite svrhe, kao što su vrijednosnice, financijski instrumenti i nekretnine. Iako se smatraju investicijskim instrumentom, ili mehanizmom, sami po sebi nisu vrijednosnice. Svi investicijski fondovi su bez obzira na tip i specijalizaciju, profesionalno upravljane grupe vrijednosnica organizirane kao jedan ili više portfelja. Kad ulagač ulaže svoja sredstva u investicijski fond, on ne ulaže ni u jednu posebnu vrijednosnicu koju fond drži, već u sve njih zajedno i to na način da kupuje ulog u fondu. Ulaganjem u investicijske fondove ulagači svoja sredstva učinkovito raspoređuju na desetak, a često i na stotinjak različitih vrijednosnica, čime značajno umanjuju rizik

svog ulaganja dok su još uvijek u poziciji dobivanja vrlo dobre stope povrata na ulaganje. (Cvjetičanin, M. , 2005, str. 125-125)

Radi lakšeg shvaćanja investicijskih fondova, oni se dijele prema njihovom tipu. Navedena podjela vidljiva je na slici 12.



Slika 12. Podjela investicijskih fondova po tipu (Cvjetičanin, M. , 2005, str. 127)

Sa slike 12. je očito da se investicijski fondovi po tipu dijele na: (Cvjetičanin, M. , 2005, str. 127-150):

- **Uzajamne** – investicijska poduzeća koja prikupljaju sredstva ulagača prodajom svojih dionica i onda tako prikupljena sredstva putem svog portfelja ulažu u različite vrijednosnice. Mogu biti:
    - **Otvoreni** – investicijske tvrtke čije dionice kupuju ulagači i nakon određenog vremena mogu prodati te dionice nazad otvorenim fondovima. Dakle, fondovi su otvoreni i prodaji, i ponovnoj kupnji.
    - **Zatvoreni** – zatvorena investicijska poduzeća koja djeluju s točno određenim, fiksnim brojem dionica, te regularno ne izdaju svoje nove dionice. Dakle, fondovi ne otkupljuju svoje dionice od ulagača, već ih ulagači i kupuju i prodaju.
- **Jednične ulagačke** – poznati i kao ograničena ulagačka partnerstva. Predstavljaju interes koji ulagači imaju u određenim stalnim grupama vrijednosnica.
- **Specijalizirane** – različiti fondovi za arbitražu i otkup poduzeća.

5.1.2.Investiranje u nekretnine

Nekretnine su poput ostalih vrsta vlasničkih ulaganja, kao dionice, gdje postoji vlasnički udio u imovini. Iako postoji potencijal za značajnu dobit, ulagač mora prihvatiti i veći rizik. Nekretnine nisu jednostavan način za steći bogatsvo, i treba biti strpljiv prilikom ulaganja. Poput dionica, nekretnine prolaze kroz dobra i loša razdoblja. Većina ljudi koji zarađuju ulagajući u nekretnine čine to zato što ulažu i drže imovinu dugi niz godina dok njihova vrijednost ne naraste. Velika većina ljudi koji ne zarađuju na nekretninama čine grešku jer prodaju nekretnine prerano, te gube previše pokušavajući izbjeći rizike. (Tyson, E., 2017, str. 217)

Mnogi ljudi svoje bogatstvo grade ulažući u nekretnine. Neki se ljudi usredotočuju isključivo na ulaganja u nekretnine, ali mnogi drugi svoje bogatstvo grade kroz tvrtke koje su osnovali ili putem drugih mogućnosti, a zatim se orijentiraju na ulaganja u nekretnine. Nekretnine, kao i sva ulaganja, imaju svoje prednosti i nedostatke. Investiranje u nekretnine je izazov i napredak u radu s ulaganjima. Vrijednosti nekretnina trenutno se kreću u korak s drugim ulaganjima, poput dionica ili ulaganja u mala poduzeća, pa je to koristan alat za napredak i poboljšavanje prihoda. (Tyson, E., 2017, str. 218)

5.1.2.1. Porast vrijednosti i prihodi

Glavni razlog zašto su nekretnine popularna metoda investiranja je taj što se od njih može zaraditi na dva glavna načina: (Tyson, E., 2017, str. 219)

- **Porast vrijednosti** – ulagač očekuje da će tijekom godina njegovo ulaganje u nekretnine imati veću vrijednost. Cijene nekretnina ne oporezuju se tijekom godina vlasništva. Ne plaća se porez na ovu dobit dok se ne proda imovinu - a čak se i tada dobitak može prebaciti u drugu investicijsku nekretninu kako bi se izbjeglo plaćanje poreza. Ako se odluči jednostavno uzeti svu dobit, a ne ju prenamijeniti, na snagu stupa federalna porezna stopa na dobit od imovine koja se drži dulje od jedne godine - poznata kao dugoročna kapitalna dobit - oporezuje se s ne više od 20 posto. To je korisno ako se uzme u obzir da se uobičajeni dohodak može oporezivati stopama koje se približavaju 40 posto.

- **Prihodom** – ulagač se nada i očekuje da će zaraditi od tekućeg posla koji vodi - iznajmljivanja imovine. Iznajmljivanje investicijske nekretnine se radi sa svrhom ostvarivanja zarade na temelju prihoda od najma nekretnine koji premašuju njegove troškove. Ako se ne naplati velika kapara, mjesečna operativna dobit obično je mala u prvim godinama vlasništva nad nekretninama. S vremenom se operativna dobit, koja podliježe uobičajenom porezu na dohodak, povećava s rastom troškova. Tijekom blagih razdoblja promjena u lokalnom gospodarstvu, najamnine mogu rasti sporije od troškova.

Kao što je prikazano u prethodnom dijelu rada, prihod od investiranja u nekretnine se sve češće ostvaruje iznajmljivanjem, te se na taj način vraća uloženo i ostvaruje velika dobit. Kao i kod bilo kojeg drugog investiranja ključno je strpljene.

5.2. Dvadeset pravila za uspješno investiranje

Za uspješno ulaganje u bilo koju granu ne postoje točno određena pravila koja vode do ostvarivanja prihoda. Međutim, Tyson (2017) navodi sljedećih dvadeset pravila koja poboljšavaju mogućnost ostvarivanja prihoda i uspješnosti investiranja: (Tyson, E., 2017)

1. Štednja je preduvjet za ulaganje,
2. Poznavanje i ulaganje u dionice, nekretnine i mala poduzeća vodi k izgradnji bogatstva,
3. Potrebna je realističnost pri očekivanju prihoda,
4. Potrebno je razmišljati dugoročno,
5. Ulaganje treba ispravno tempirati s obzirom na mogućnosti,
6. Raznolikost ulaganja,
7. Potrebno je sagledati širu sliku,
8. Zanemariti sitnice,
9. Potrebno je ispravno razdijeliti imovinu,
10. Istražiti tržište prije ulaganja,
11. Pripaziti na poreze,
12. Uzeti u obzir vlastite vještine i mogućnosti,
13. Minimizirati naknade, gdje je to moguće,

14. Pokoriti i vladati tržištem je teško ostvarivo,
15. Držati se zacrtanog plana i ne odustajati kada stvari nisu najbolje,
16. Ignorirati prognostičare,
17. Minimizirati razmjene,
18. Zaposliti iskusnog savjetnika,
19. Neprestano se educirati i istraživati tržište
20. Zdravlje i osobni život su na prvom mjestu i mnogo su važniji od financijskog stanja.

5.3. Primjer investiranja

Radi lakšeg razumijevanja, prikazan je i sljedeći primjer koji je izrađen temeljem stvarnih podataka tvrtke Amazon. U sljedećoj tablici prikazani su podaci za tri razdoblja u ovoj godini koji je osnova za prikaz na stvarnom primjeru i omogućuje bolje razumijevanje kako funkcionira postupak investiranja. Podaci se odnose na kretanje vrijednosti dionica navedene tvrtke.

Tablica 1. Kretanje dionica tvrtke Amazon

Datum	Posljednja	Volumen	Početna	Najviša	Najniža
30/04/2021	$3467,42	7,009,346	$3525,12	$3554	$3462,5
23/04/2021	$3340,88	3,196,652	$3319,1	$3375	$3308,5
16/04/2021	$3399,44	3,186,049	$3380	$3406,8	$3355,59

(Izvor: AMZN Historical Data, 2021)

Koristeći dane podatke bit će prikazan najprije primjer isplativog, a zatim neisplativog ulaganja:

- **Isplativo ulaganje**: Ako je određena osoba kupila 100 dionica tvrtke Amazon 23. travnja 2021. godine po najnižoj cijeni - $3308,5, navedeno je platila $330.850. Prateći razvijanje tržišta, osoba je uvidjela da je tjedan dana kasnije, odnosno 30. travnja 2021., cijena dionica na zatvaranju bila $3467,42 i odlučila je prodati svojih 100 dionica. Za navedeno osoba dobija $346.742 i ostvaruje dobit od $15,892 ($346.742 - $330.850) u samo tjedan dana.

- **Neisplativo ulaganje**: Ako je određena osoba kupila 100 dionica tvrtke Amazon 16. travnja 2021. godine po početnoj cijeni - $3380, navedeno je platila $338.000. Bez velikog praćenja tržišta i kretanja cijena, osoba odlučuje prodati svoje dionice 23. travnja 2021. po posljednjo cijeni - $3340,88 . Za navedeno osoba dobija $334.088 i ostvaruje gubitak od $3,912 ($334.088 - $338.000) zbog lošeg donošenja odluka i praćenja kretanja cijena.

Iz navedenih primjera može se zaključiti da je kod investiranja vrlo bitno pratiti kretanje cijena i situacije na tržištu, kao i izbjegavati brzanje u donošenju odluka jer je investiranje, kako je već rečeno dugotrajan proces proučavanja i analiziranja.

6. Problem investiranja

U nastavku rada definiran je sam problem investiranja, kao i njegova primjena na realnom primjeru. Također, prikazani su računski postupci, kao i grafički prikaz istog.

6.1. Definiranje problema

Za problem investiranja se može reći da je to tipičan primjer problema dinamičkog programiranja koji ne sadrži vremensku komponentu. No, kod njega se može umjetno izvršiti dekompozicija na više etapa, što odgovara nizu aktivnosti. Nakon obavljene dekompozicije može se primijeniti načelo optimizacije metodom dinamičkog programiranja. (Nemhauser, G.L.,1966, str. 28)

Programirana situacija uključuje izvjesnu količinu ekonomskih resursa, kao na primjer strojeve, novac, ili ljude koji moraju biti raspoređeni na određeni broj različitih aktivnosti, odnosno etapa. Situacija, kako je već rečeno, ne uključuje vremensku komponentu. (Fabricky, W.J., i Torgersen, P. , 1966, str. 422).

Problem s programiranom situacijom nastaje zbog brojnih načina na koje se može obaviti raspored resursa na određeni broj aktivnosti. Rasporedom cijelog, ili samo dijela, resursa na određenu aktivnost dobiva se izvjestan prihod. Koliki će biti prihod ovisi o specifičnosti same aktivnosti i količine resursa raspoređenog na upravo tu aktivnost. Pretpostavka je da se prihodi postignuti na različitim aktivnostima mogu mjeriti zajedničkom jedinicom mjere te da je prihod neke aktivnosti nezavisan od rasporeda ostatka resursa na druge aktivnosti. Pretpostavka je i da je ukupan prihod zbroj svih pojedinačnih prihoda. Cilj kojem se teži je rasporediti raspoloživi resurs na različite aktivnosti tako da se maksimizira ukupni prihod. (Gillett, B.E., 1976, str. 12)

6.2. Izgradnja modela

Broj aktivnosti označen je s N, odbrojavanjem aktivnosti označeno je sa $i = 1, 2, \dots, N$. Svaka aktivnost ima pridruženu svoju funkciju prihoda koja prikazuje zavisnost prihoda o aktivnosti na koju je raspoređena količina resursa. Količina resursa raspoređena na aktivnost (i), označena je s $x(i)$. Prema tome, funkcija prihoda označava se s $g(i)[\, x(i)\,]$. S obzirom da se pretpostavlja da su aktivnosti nezavisne i

da su prihodi svih aktivnosti zbrojivi, ukupan prihod se prikazuje kao $R\,[\,x(1), x(2), \ldots, x(N)\,] = g(1)[\,x(1)\,] + g(2)[\,x(2)] + \cdots + g(N)[\,x(N)]$. (Dobrenić, S., 1978, str. 326)

Ograničena količina resursa Q dovodi do ograničenja $Q = x(1) + x(2) + \cdots + x(N)$, gdje je $x(i) \geq 0$. Cilj je maksimizirati ukupni prihod R za sve $x(i)$ uvjetovane ograničenjem Q. Problem maksimizacije ukupnog prihoda procesa alokacije se promatra kao jedan član obitelji procesa raspoređivanja. Pretpostavka je da količina resursa Q, i broj aktivnosti N nisu fiksni, tako da mogu poprimiti bilo koju vrijednost, uz uvjet da, naravno, N mora biti cijeli broj. Za to ograničenje možemo reći da je zapravo fiktivno jer će se raspoređivanja obavljati na jednoj od aktivnosti. U prijevodu, određena količina resursa je raspoređena najprije na $N - tu$ aktivnost, a zatim na $N - 1$ aktivnost... (Dobrenić, S., 1978, str. 326)

S obzirom da maksimizacija ukupnih prihoda R ovisi o količini resursa Q, i broju aktivnosti N, zavisnost mora biti eksplicitno postavljena nizom funkcija:

$$f(N)[\,Q\,] = \max_{x(i)} R\,[\,x(1), x(2), \ldots, x(N)]$$

Funkcija $f(N)[\,Q\,]$ prikazuje maksimum prihoda postignutog raspoređivanjem resursa Q na $N - tu$ aktivnost. Ako se pretpostavi da je $g(i)[\,0\,] = 0$ za svaku aktivnost, evidentno je da je tada i rezultat funkcije $f(N)$ jednak 0. Zatim se može pretpostaviti i da je $f(1)[\,Q\,] = g(1)[\,Q\,]$. Ove dvije pretpostavke zapravo izražavaju prihod koji se očekuje od $N - te$ aktivnosti, ako resursi nisu raspoređeni i prihod koji se očekuje od prve aktivnosti, ako su svi resursi raspoređeni na nju. (Dobrenić, S., 1978, str. 327)

Funkcionalna relacija, koja povezuje funkcije $f(N)[\,Q]$ i $f(N-1)[\,Q]$ za proizvoljne vrijednosti za količinu resursa Q, i broj aktivnosti N, može se razviti po sljedećim koracima: (Dobrenić, S., 1978, str. 327)

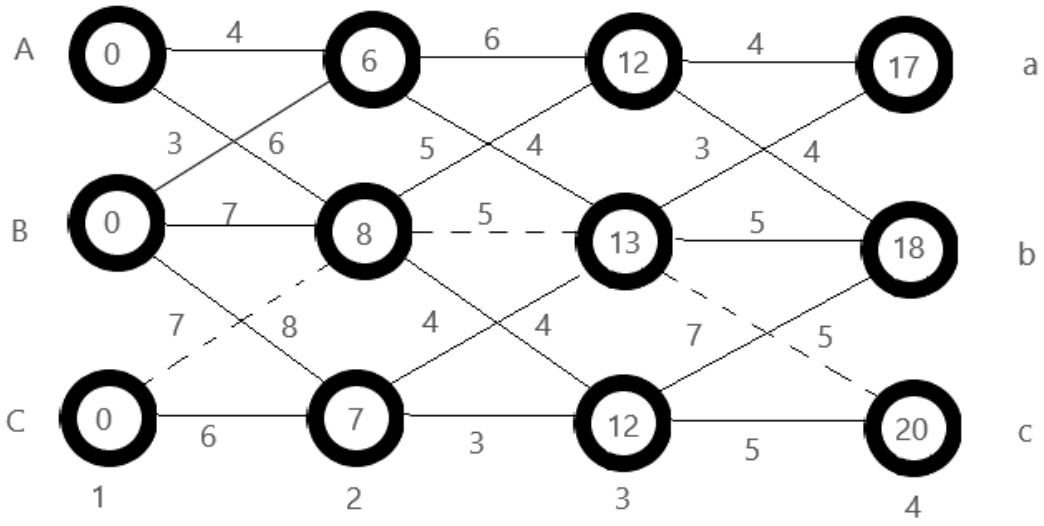
- Ako je $x\,(N)$, uz ograničenje $0 \leq x\,(N) \leq Q$, raspored resursa za $N - tu$ aktivnost, tada će s obzirom na vrijednost $x\,(N)$, ostati **količina resursa $\boldsymbol{Q - x(N)}$,**
- **Prihod aktivnosti** $(\boldsymbol{N-1})$ može se izraziti kao $f(N-1)[\,Q - x(N)]$,
- **Ukupan prihod** procesa s N aktivnosti može se izraziti kao $g(N)[\,x(N)] + f(N-1)[\,Q - x(N)\,]$,
- **Optimalni izbor** $x(N)$ je onaj koji maksimizira ukupan prihod,

- **Osnovni model dinamičkog programiranja za problem investiranja je:** $f(N)[\,Q\,] = \max\,\{g(N)[\,x(N)] + f(N-1)[\,Q - x(N)]\}, N = 2,3, \ldots$
  $0 \le x(N) \le Q$

6.3. Računski postupak

Osnovni model dinamičkog programiranja za problem investiranja daje metodu za dobivanje niza $f(N)[\,Q\,]$ kada je jednom poznat $f\,(1)[\,Q\,]$. Budući da ta funkcija determinira $f\,(2)[\,Q\,]$, sigurno je i da $f\,(2)[\,Q\,]$ dovodi do određivanja $f\,(3)[\,Q\,]$... Ovaj povratni odnos razvija se u smislu da $f\,(N-1)[\,Q\,]$ determinira $f\,(N)[\,Q\,]$ u vremenu u kojem se proces zaustavlja. (Dobrenić, S., 1978, str. 328)

Za lakše razumijevanje tog postupka, predstavljen je sljedeći mrežni dijagram dinamičkog programiranja.



Slika 13. Prikaz dinamičkog programiranja pomoću mrežnog dijagrama (Fabricky, W.J., i Torgersen, P. , 1966, str. 423).

Dakle, na slici je vidljiv poželjan maksimalni put kroz mrežni dijagram. Problem se može riješiti utvrđivanjem svih mogućih puteva i izračunavanjem vrijednosti za svaki. Međutim, posljedica ove metode može biti izostavljanje nekog puta, a uz to zahtjeva i mnogo računanja. Jedno od mogućih rješenja ovog problema je da proces promatramo kroz korištenje metode dinamičkog programiranja i pretpostavimo da se radi o višeetapnom procesu odlučivanja.

Maksimalni put od etape 1 do etape 2 javlja se za svaku krajnju točku etape 2. Maksimalni put se javlja bez obzira na izbor početne točke, jer egzistira put do krajnje točke koji mora biti maksimum za optimalnu dvoetapnu politiku. Na taj način je bez prevelikog nepotrebnog računanja određena optimalna politika. Isto se odvija za troetapnu, četvoretapnu politiku i tako dalje. Nadalje treba razmotriti situaciju rasporeda resursa. Potpuni skup vrijednosti za $f(N)[\,Q\,]$ u području od 0 do Q može se pronaći uz pretpostavku konačnog broja točaka mreže. Maksimizacija se može obaviti kada je $N = 1$, budući da je tada $f(1)[\,Q\,] = g(1)[\,Q\,]$. Možemo izračunati $f(1)[\,Q\,]$. Korištenjem osnovnog modela dinamičkog programiranja za problem investiranja dalje računamo $f(2)[\,Q\,]$ za $N = 2$ tako da je: (Dobrenić, S., 1978, str. 330)

$$f(2)[\,Q\,] = \max\{\, g(2)[\,x(2)] + f(1)[\,Q - x(2)]\}$$
$$0 \le x(2) \le Q$$

Maksimizacija procesa počinje određivanjem vrijednosti $g(2)[\,0\,] + f(1)[\,Q\,]$ i $g(2)[\,\Delta\,] + f(1)[\,Q - \Delta\,]$. Veća od ovih vrijednosti se pamti i uspoređuje s vrijednosti $g(2)[\,2\,\Delta\,] + f(1)[\,Q - 2\,\Delta\,]$. Kao i ranije, veća od ovih vrijednosti se pamti i uspoređuje s $g(2)[\,3\,\Delta\,] + f(1)[\,Q - 3\,\Delta\,]$. Ovaj proces se nastavlja sve dok se ne razmotre sve vrijednosti Δ. Rezultat se prikazuje kao tablica vrijednosti koja se može pridružiti do N etapa, uključujući sve potrebne vrijednosti. Izgled tablice je kako slijedi. (Dobrenić, S., 1978, str. 330)

Tablica 2. Prikaz sheme računanja

Q	$X(1)$	$f(1)[\,Q\,]$	$X(2)$	$f(2)[\,Q\,]$	.....
0					
Δ					
2 Δ					
3 Δ					
.....					

(Izvor: Dobrenić, S., 1978, str. 330)

Navedena tablica služi za izračunavanje svih potrebnih vrijednosti, ovisno o ulaznim parametrima. Također, tablicu je potrebno koristiti za upisivanje i lakše praćenje vrijednosti, kako će biti prikazano kasnije u radu.

6.4. Grafički prikaz ovisnosti dobiti o investicijama

Poznato je da se uspješnim i promišljenim ulaganjem u neku djelatnost ostvaruje dobit. U općem slučaju dobit raste s veličinom uloženog kapitala dok ne dođe do zasićenja. Da bi se prikazala dobit ovisno o uloženim sredstvima, koristi se metoda dinamičkog programiranja. (Kalpić, D., i Mornar, V., 1996, str. 102)

U prethodnom potpoglavlju objašnjem je proces korištenja metode dinamičkog programiranja. Za dobivenu tablicu rješenja prikazanu na kraju navedenog poglavlja može se izraditi i grafički prikaz za bolju vizualizaciju ovisnosti dobiti i investicija.

Za izradu grafičkog prikaza koristit će se primjer koji pruža Kaufmann (1970):

„Pretpostavimo da raspolažemo kapitalom od 10 novčanih jedinica koji treba uložiti u neku proizvodnju u 4 područja. Ekonomske analize pokazuju koliku bi dobit trebalo u pojedinim područjima očekivati uz određeno investiranje sredstava. Za svako pojedino područje A, B, C i D očekivana dobit prikazana je tablicom:“

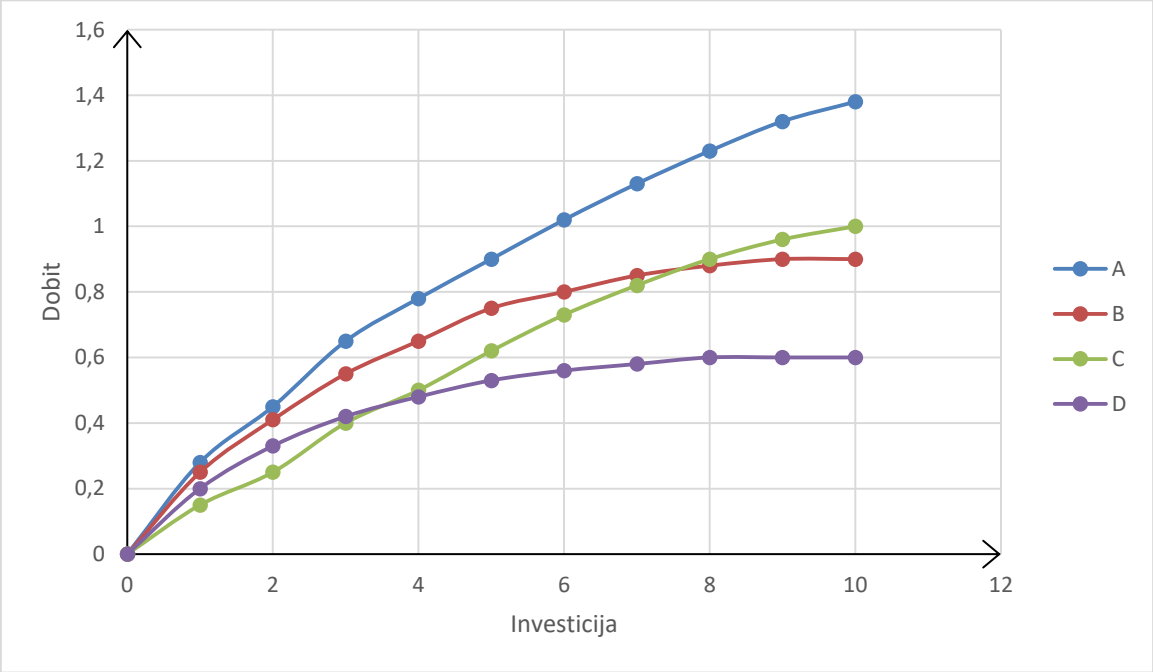
Tablica 3. Očekivana dobit za pojedina područja

Investicija	Dobit A	Dobit B	Dobit C	Dobit D
0	0	0	0	0
1	0.28	0.25	0.15	0.20
2	0.45	0.41	0.25	0.33
3	0.65	0.55	0.40	0.42
4	0.78	0.65	0.50	0.48
5	0.90	0.75	0.62	0.53
6	1.02	0.80	0.73	0.56
7	1.13	0.85	0.82	0.58
8	1.23	0.88	0.90	0.60
9	1.32	0.90	0.96	0.60
10	1.38	0.90	1	0.60

(Izvor: Kaufmann, A. 1970)

U ovom dijelu rada zadani primjer biti će prikazan samo grafički, odnosno u ovom slučaju za tablicu 4. Ostatak, odnosno cijeli računski proces za rješavanje problema investiranje bit će prikazan u narednom poglavlju.

Za Kauffmanov primjer, ovisnost dobiti o investicijama, prikazana grafički izgleda kako slijedi:



Slika 14. Ovisnost dobiti o investicijama (Kaufmann, 1970)

Iz Slike 14. vidljivo je da vrijednost pojedine dobiti ne ovisi o drugoj dobit, već svaka dobit direktno ovisi samo o investicijama. Za svaku vrijednost investicije, postoje četiri različite dobiti kako je definirano u tablici 3., a ovdje je prikazano njihovo kretanje i prikaz dobiti, radi lakšeg shvaćanja i vizualizacije ovisnosti dobiti i investicija. Po kretanjima na grafu, može se zaključiti da što se više uloži, odnosno što se više riskira, to se više i dobije. Također, što je manja investicija to je i manja dobit.

7. Primjer problema investiranja

Primjer problema investiranja koji će biti razrađen temeljen je na problemu postavljenom u knjizi prof. Dobrenića (Dobrenić, S., 1978, str. 331). U nastavku je prvo postavljen opis problema koji će se rješavati, zatim je priložen sam postupak rješenja, kao i grafički prikaz.

7.1.1. Opis problema

Tvrtka raspolaže s deset jedinica novca (100 mil. kuna). Taj iznos se može investirati u nekoliko rata. Investirati se može u četiri različite aktivnosti. Funkcije neto prihoda za svaku aktivnost dane su tablicom 5. ispod. Neto prihod svake aktivnosti je nezavisan od rasporeda sredstava na druge aktivnosti. Također je ukupan neto prihod zbroj pojedinačnih neto prihoda. Svaka aktivnost izražava funkciju neto prihoda koja raste za dio njezinog graničnog iznosa, isprva u rastućem, a zatim u opadajućem omjeru. To dolazi od zakona opadajućeg prihoda, što je tipično za mnoge aktivnosti. Konačna mreža točaka za ovaj problem je postavljena diskretnom prirodom funkcija neto prihoda. Bilo bi moguće direktno odbrojavanje svih načina na koji se može određeni broj rata sredstava investirati u četiri različite aktivnosti. Međutim, može se koristiti model dinamičkog programiranja kako bi se uštedjelo u radu na računanju.

Sljedeći primjer je napravljen kao vlastita ideja. Većinski tekst zadatka se temelji na zadatku kojeg Dobrenić, S., navodi, a prikazan je u tekstu iznad. Ovaj primjer se rješava radi dodatnih funkcionalnosti i ispitivanja metode diamičkog programiranja za problem investiranja na većem skupu podataka. Ovaj primjer poslužit će u daljnjim poglavljima kao neka vrsta testnog slučaja za programsko rješenje čiji će prikaz implementacije biti pružen u sljedećim poglavljima.

Zadatak se zapravo temelji na nekoliko osnovnih postavki koje vrijedi istaknuti:

- Tvrtka raspolaže s **deset jedinica – 100 mil. kuna**
- Investirati se može u **četiri** različite aktivnosti
- Funkcije neto prihoda za svaku aktivnost dane su u **sljedećoj** tablici:

Tablica 4. Funkcije neto prihoda – vlastiti primjer

Q	$g(1)[\,Q\,]$	$g(2)[\,Q\,]$	$g(3)[\,Q\,]$	$g(4)[\,Q\,]$
0	0	0	0	0
1	10	8	6	4
2	17	14	12	10
3	25	22	23	18
4	33	28	33	30
5	36	32	37	35
6	41	39	45	43
7	50	46	51	49
8	53	51	56	55
9	58	56	61	60
10	66	61	65	64

(Izvor: Vlastiti rad, 2021)

7.1.2. Računsko rješenje problema

Neto prihod koji se očekuje od prve aktivnosti ako se sva raspoloživa sredstva rasporede na nju računaju se po formuli:

$$f(1)[\,Q\,] = g(1)[\,Q\,]$$

Koristeći navedenu formulu, i vrijednosti prve aktivnosti u tablici, slijedi:

$$f(1)[\,0\,] = g(1)[\,0\,] = 0$$

$$f(1)[\,1\,] = g(1)[\,1\,] = 10$$

$$f(1)[\,2\,] = g(1)[\,2\,] = 17$$

$$f(1)[\,3\,] = g(1)[\,3\,] = 25$$

$$f(1)[\,4\,] = g(1)[\,4\,] = 33$$

$$f(1)[\,5\,] = g(1)[\,5\,] = 36$$

$$f(1)[\,6\,] = g(1)[\,5\,] = 41$$

$$f(1)[\,7\,] = g(1)[\,5\,] = 50$$

$$f(1)[\,8\,] = g(1)[\,5\,] = 53$$

$$f(1)[\,9\,] = g(1)[\,5\,] = 58$$

$$f(1)[\,10\,] = g(1)[\,5\,] = 66$$

Dobivene vrijednosti zatim uvrštavamo u tablicu rješenja:

Tablica 5. Rješenje problema investiranja 1/4 - vlastiti primjer

Q	$X(1)$	$f(1)[\,Q\,]$	$X(2)$	$f(2)[\,Q\,]$	$x(3)$	$f(3)[\,Q\,]$	$x(4)$	$f(4)[\,Q\,]$
0	0	0						
1	1	10						
2	2	17						
3	3	25						
4	4	33						
5	5	36						
6	6	41						
7	7	50						
8	8	53						
9	9	58						
10	10	66						

(Izvor: Vlastiti rad, 2021)

Koristeći rezultate za $f(1)[\ Q\ ]$, izračunat ćemo vrijednosti za $f(2)[\ Q\ ]$, preko formule osnovnog modela dinamičkog programiranja za problem investiranja:

$$f(N)[\ Q\ ] = \max_{0 \le x(N) \le Q} \{g(N)[\ x(N)] + f(N-1)[\ Q - x(N)]\}, N = 2,3, \ldots$$

Za $Q = 0$:

$$f(2)\ [\ 0\ ] = 0$$

Za $Q = 1$:

$$f(2)[\ 1\ ] = \max_{0 \le x(2) \le 1} \begin{Bmatrix} g(2)[0] + f(1)[1] = 0 + 10 = 10 \\ g(2)[1] + f(1)[0] = 8 + 0 = 8 \end{Bmatrix} \rightarrow f(2) = 10, x(2) = 0$$

Za $Q = 2$:

$$f(2)[\ 2\ ] = \max_{0 \le x(2) \le 2} \begin{Bmatrix} g(2)[0] + f(1)[2] = 0 + 17 = 17 \\ g(2)[1] + f(1)[1] = 8 + 10 = 18 \\ g(2)[2] + f(1)[0] = 14 + 0 = 14 \end{Bmatrix} \rightarrow f(2) = 18, x(2) = 1$$

Za $Q = 3$:

$$f(2)[\ 3\ ] = \max_{0 \le x(2) \le 3} \begin{Bmatrix} g(2)[0] + f(1)[3] = 0 + 25 = 25 \\ g(2)[1] + f(1)[2] = 8 + 17 = 25 \\ g(2)[2] + f(1)[1] = 14 + 10 = 24 \\ g(2)[3] + f(1)[0] = 22 + 0 = 22 \end{Bmatrix} \rightarrow f(2) = 25, \begin{matrix} x(2) = 0 \\ x(2) = 1 \end{matrix}$$

Za $Q = 4$:

$$f(2)[\ 4\ ] = \max_{0 \le x(2) \le 4} \begin{Bmatrix} g(2)[0] + f(1)[4] = 0 + 33 = 33 \\ g(2)[1] + f(1)[3] = 8 + 25 = 33 \\ g(2)[2] + f(1)[2] = 14 + 17 = 31 \\ g(2)[3] + f(1)[1] = 22 + 10 = 32 \\ g(2)[4] + f(1)[0] = 28 + 0 = 28 \end{Bmatrix} \rightarrow f(2) = 33, \begin{matrix} x(2) = 0 \\ x(2) = 1 \end{matrix}$$

Za $Q = 5$:

$$f(2)[\ 5\ ] = \max_{0 \le x(2) \le 5} \begin{Bmatrix} g(2)[0] + f(1)[5] = 0 + 36 = 36 \\ g(2)[1] + f(1)[4] = 8 + 33 = 41 \\ g(2)[2] + f(1)[3] = 14 + 25 = 39 \\ g(2)[3] + f(1)[2] = 22 + 17 = 39 \\ g(2)[4] + f(1)[1] = 28 + 10 = 38 \\ g(2)[5] + f(1)[0] = 32 + 0 = 32 \end{Bmatrix} \rightarrow f(2) = 41, x(2) = 1$$

Za $Q = 6$:

$$f(2)[\,6\,] = \max_{0 \le x(2) \le 6} \begin{Bmatrix} g(2)[0] + f(1)[6] = 0 + 41 = 41 \\ g(2)[1] + f(1)[5] = 8 + 36 = 44 \\ g(2)[2] + f(1)[4] = 14 + 33 = 47 \\ g(2)[3] + f(1)[3] = 22 + 25 = 47 \\ g(2)[4] + f(1)[2] = 28 + 17 = 45 \\ g(2)[5] + f(1)[1] = 32 + 10 = 41 \\ g(2)[6] + f(1)[0] = 39 + 0 = 39 \end{Bmatrix} \rightarrow f(2) = 47, \begin{matrix} x(2) = 2 \\ x(2) = 3 \end{matrix}$$

Za $Q = 7$:

$$f(2)[\,7\,] = \max_{0 \le x(2) \le 7} \begin{Bmatrix} g(2)[0] + f(1)[7] = 0 + 50 = 50 \\ g(2)[1] + f(1)[6] = 8 + 41 = 49 \\ g(2)[2] + f(1)[5] = 14 + 36 = 50 \\ g(2)[3] + f(1)[4] = 22 + 33 = 55 \\ g(2)[4] + f(1)[3] = 28 + 25 = 53 \\ g(2)[5] + f(1)[2] = 32 + 17 = 49 \\ g(2)[6] + f(1)[1] = 39 + 10 = 49 \\ g(2)[7] + f(1)[0] = 46 + 0 = 46 \end{Bmatrix} \rightarrow f(2) = 55, x(2) = 3$$

Za $Q = 8$:

$$f(2)[\,8\,] = \max_{0 \le x(2) \le 8} \begin{Bmatrix} g(2)[0] + f(1)[8] = 0 + 53 = 53 \\ g(2)[1] + f(1)[7] = 8 + 50 = 58 \\ g(2)[2] + f(1)[6] = 14 + 41 = 55 \\ g(2)[3] + f(1)[5] = 22 + 36 = 58 \\ g(2)[4] + f(1)[4] = 28 + 33 = 61 \\ g(2)[5] + f(1)[3] = 32 + 25 = 57 \\ g(2)[6] + f(1)[2] = 39 + 17 = 56 \\ g(2)[7] + f(1)[1] = 46 + 10 = 56 \\ g(2)[8] + f(1)[0] = 51 + 0 = 51 \end{Bmatrix} \rightarrow f(2) = 61, x(2) = 4$$

Za $Q = 9$:

$$f(2)[\,9\,] = \max_{0 \le x(2) \le 9} \begin{Bmatrix} g(2)[0] + f(1)[9] = 0 + 58 = 58 \\ g(2)[1] + f(1)[8] = 8 + 53 = 61 \\ g(2)[2] + f(1)[7] = 14 + 50 = 64 \\ g(2)[3] + f(1)[6] = 22 + 41 = 63 \\ g(2)[4] + f(1)[5] = 28 + 36 = 64 \\ g(2)[5] + f(1)[4] = 32 + 33 = 65 \\ g(2)[6] + f(1)[3] = 39 + 25 = 64 \\ g(2)[7] + f(1)[2] = 46 + 17 = 63 \\ g(2)[8] + f(1)[1] = 51 + 10 = 61 \\ g(2)[9] + f(1)[0] = 56 + 0 = 56 \end{Bmatrix} \rightarrow f(2) = 65, x(2) = 5$$

Za $Q = 10$:

$$f(2)[\,10\,] = \max_{0 \le x(2) \le 10} \begin{Bmatrix} g(2)[0] + f(1)[10] = 0 + 66 = 66 \\ g(2)[1] + f(1)[9] = 8 + 58 = 66 \\ g(2)[2] + f(1)[8] = 14 + 53 = 67 \\ g(2)[3] + f(1)[7] = 22 + 50 = 72 \\ g(2)[4] + f(1)[6] = 28 + 41 = 69 \\ g(2)[5] + f(1)[5] = 32 + 36 = 68 \\ g(2)[6] + f(1)[4] = 39 + 33 = 69 \\ g(2)[7] + f(1)[3] = 46 + 25 = 71 \\ g(2)[8] + f(1)[2] = 51 + 17 = 68 \\ g(2)[9] + f(1)[1] = 56 + 10 = 66 \\ g(2)[10] + f(1)[0] = 61 + 0 = 61 \end{Bmatrix} \rightarrow \begin{matrix} f(2) = 72 \\ x(2) = 3 \end{matrix}$$

Dobivene vrijednosti zatim uvrštavamo u tablicu rješenja:

Tablica 6. Rješenje problema investiranja 2/4 - vlastiti primjer

Q	$X(1)$	$f(1)[\,Q\,]$	$X(2)$	$f(2)[\,Q\,]$	$x(3)$	$f(3)[\,Q\,]$	$x(4)$	$f(4)[\,Q\,]$
0	0	0	0	0				
1	1	10	0	10				
2	2	17	1	18				
3	3	25	0, 1	25				
4	4	33	0, 1	33				
5	5	36	1	41				
6	6	41	2, 3	47				
7	7	50	3	55				
8	8	53	4	61				
9	9	58	5	65				
10	10	66	3	72				

(Izvor: Vlastiti rad, 2021)

Koristeći rezultate za $f(2)[\ Q\ ]$, izračunat ćemo vrijednosti za $f(3)[\ Q\ ]$, preko već poznate formule osnovnog modela dinamičkog programiranja za problem investiranja:

Za $Q = 0$:

$$f(3)\ [\ 0\ ] = 0$$

Za $Q = 1$:

$$f(3)[\ 1\ ] = \max_{0 \le x(3) \le 1} \begin{Bmatrix} g(3)[0] + f(2)[1] = 0 + 10 = 10 \\ g(3)[1] + f(2)[0] = 6 + 0 = 6 \end{Bmatrix} \rightarrow f(3) = 10, x(3) = 0$$

Za $Q = 2$:

$$f(3)[\ 2\ ] = \max_{0 \le x(3) \le 2} \begin{Bmatrix} g(3)[0] + f(2)[3] = 0 + 18 = 18 \\ g(3)[1] + f(2)[1] = 6 + 10 = 16 \\ g(3)[2] + f(2)[0] = 12 + 0 = 12 \end{Bmatrix} \rightarrow f(3) = 18, x(3) = 0$$

Za $Q = 3$:

$$f(3)[\ 3\ ] = \max_{0 \le x(3) \le 3} \begin{Bmatrix} g(3)[0] + f(2)[3] = 0 + 25 = 25 \\ g(3)[1] + f(2)[2] = 6 + 18 = 24 \\ g(3)[2] + f(2)[1] = 12 + 10 = 22 \\ g(3)[3] + f(2)[0] = 23 + 0 = 23 \end{Bmatrix} \rightarrow f(3) = 25, x(3) = 0$$

Za $Q = 4$:

$$f(3)[\ 4\ ] = \max_{0 \le x(3) \le 4} \begin{Bmatrix} g(3)[0] + f(2)[4] = 0 + 33 = 33 \\ g(3)[1] + f(2)[3] = 6 + 25 = 31 \\ g(3)[2] + f(2)[2] = 12 + 18 = 30 \\ g(3)[3] + f(2)[1] = 23 + 10 = 33 \\ g(3)[4] + f(2)[0] = 33 + 0 = 33 \end{Bmatrix} \rightarrow f(3) = 33, \begin{matrix} x(3) = 0 \\ x(3) = 3 \\ x(3) = 4 \end{matrix}$$

Za $Q = 5$:

$$f(3)[\ 5\ ] = \max_{0 \le x(3) \le 5} \begin{Bmatrix} g(3)[0] + f(2)[5] = 0 + 41 = 41 \\ g(3)[1] + f(2)[4] = 6 + 33 = 39 \\ g(3)[2] + f(2)[3] = 12 + 25 = 37 \\ g(3)[3] + f(2)[2] = 23 + 18 = 41 \\ g(3)[4] + f(2)[1] = 33 + 10 = 43 \\ g(3)[5] + f(2)[0] = 37 + 0 = 37 \end{Bmatrix} \rightarrow f(3) = 43, x(3) = 4$$

Za $Q = 6$:

$$f(3)[\,6\,] = \max_{0 \le x(3) \le 6} \left\{ \begin{array}{c} g(3)[0] + f(2)[6] = 0 + 47 = 47 \\ g(3)[1] + f(2)[5] = 6 + 41 = 47 \\ g(3)[2] + f(2)[4] = 12 + 33 = 45 \\ g(3)[3] + f(2)[3] = 23 + 25 = 48 \\ g(3)[4] + f(2)[2] = 33 + 18 = 51 \\ g(3)[5] + f(2)[1] = 37 + 10 = 47 \\ g(3)[6] + f(2)[0] = 45 + 0 = 45 \end{array} \right\} \rightarrow f(3) = 51, x(3) = 4$$

Za $Q = 7$:

$$f(3)[\,7\,] = \max_{0 \le x(3) \le 7} \left\{ \begin{array}{c} g(3)[0] + f(2)[7] = 0 + 55 = 55 \\ g(3)[1] + f(2)[6] = 6 + 47 = 53 \\ g(3)[2] + f(2)[5] = 12 + 41 = 53 \\ g(3)[3] + f(2)[4] = 23 + 33 = 56 \\ g(3)[4] + f(2)[3] = 33 + 25 = 58 \\ g(3)[5] + f(2)[2] = 37 + 18 = 55 \\ g(3)[6] + f(2)[1] = 45 + 10 = 55 \\ g(3)[7] + f(2)[0] = 51 + 0 = 51 \end{array} \right\} \rightarrow f(3) = 58, x(3) = 4$$

Za $Q = 8$:

$$f(3)[\,8\,] = \max_{0 \le x(3) \le 8} \left\{ \begin{array}{c} g(3)[0] + f(2)[8] = 0 + 61 = 61 \\ g(3)[1] + f(2)[7] = 6 + 55 = 61 \\ g(3)[2] + f(2)[6] = 12 + 47 = 59 \\ g(3)[3] + f(2)[5] = 23 + 41 = 64 \\ g(3)[4] + f(2)[4] = 33 + 33 = 66 \\ g(3)[5] + f(2)[3] = 37 + 25 = 62 \\ g(3)[6] + f(2)[2] = 45 + 18 = 63 \\ g(3)[7] + f(2)[1] = 51 + 10 = 61 \\ g(3)[8] + f(2)[0] = 56 + 0 = 56 \end{array} \right\} \rightarrow f(3) = 66, x(3) = 4$$

Za $Q = 9$:

$$f(3)[\,9\,] = \max_{0 \le x(3) \le 9} \left\{ \begin{array}{c} g(3)[0] + f(2)[9] = 0 + 65 = 65 \\ g(3)[1] + f(2)[8] = 6 + 61 = 67 \\ g(3)[2] + f(2)[7] = 12 + 55 = 67 \\ g(3)[3] + f(2)[6] = 23 + 47 = 70 \\ g(3)[4] + f(2)[5] = 33 + 41 = 74 \\ g(3)[5] + f(2)[4] = 37 + 33 = 70 \\ g(3)[6] + f(2)[3] = 45 + 25 = 70 \\ g(3)[7] + f(2)[2] = 51 + 18 = 69 \\ g(3)[8] + f(2)[1] = 56 + 10 = 66 \\ g(3)[9] + f(2)[0] = 61 + 0 = 61 \end{array} \right\} \rightarrow f(3) = 74, x(3) = 4$$

Za $Q = 10$:

$$f(3)[\,10\,] = \max_{0 \le x(3) \le 10} \begin{Bmatrix} g(3)[0] + f(2)[10] = 0 + 72 = 72 \\ g(3)[1] + f(2)[9] = 6 + 65 = 71 \\ g(3)[2] + f(2)[8] = 12 + 61 = 73 \\ g(3)[3] + f(2)[7] = 23 + 55 = 78 \\ g(3)[4] + f(2)[6] = 33 + 47 = 80 \\ g(3)[5] + f(2)[5] = 37 + 41 = 78 \\ g(3)[6] + f(2)[4] = 45 + 33 = 78 \\ g(3)[7] + f(2)[3] = 51 + 25 = 76 \\ g(3)[8] + f(2)[2] = 56 + 18 = 74 \\ g(3)[9] + f(2)[1] = 61 + 10 = 71 \\ g(3)[10] + f(2)[0] = 65 + 0 = 65 \end{Bmatrix} \rightarrow \begin{matrix} f(3) = 80 \\ x(3) = 4 \end{matrix}$$

Dobivene vrijednosti zatim uvrštavamo u tablicu rješenja:

Tablica 7. Rješenje problema investiranja 3/4 - vlastiti primjer

Q	$X(1)$	$f(1)[\,Q\,]$	$X(2)$	$f(2)[\,Q\,]$	$x(3)$	$f(3)[\,Q\,]$	$x(4)$	$f(4)[\,Q\,]$
0	0	0	0	0	0	0		
1	1	10	0	10	0	10		
2	2	17	1	18	0	18		
3	3	25	0, 1	25	0	25		
4	4	33	0, 1	33	0, 3, 4	33		
5	5	36	1	41	4	43		
6	6	41	2, 3	47	4	51		
7	7	50	3	55	4	58		
8	8	53	4	61	4	66		
9	9	58	5	65	4	74		
10	10	66	3	72	4	80		

(Izvor: Vlastiti rad, 2021)

Koristeći rezultate za $f(3)[\ Q\ ]$, izračunat ćemo vrijednosti za $f(4)[\ Q\ ]$, preko već poznate formule osnovnog modela dinamičkog programiranja za problem investiranja:

Za $Q = 0$:

$$f(4)\ [\ 0\ ] = 0$$

Za $Q = 1$:

$$f(4)[\ 1\ ] = \max_{0 \le x(4) \le 1} \begin{Bmatrix} g(4)[0] + f(3)[1] = 0 + 10 = 10 \\ g(4)[1] + f(3)[0] = 4 + 0 = 4 \end{Bmatrix} \rightarrow f(4) = 10, x(4) = 0$$

Za $Q = 2$:

$$f(4)[\ 2\ ] = \max_{0 \le x(4) \le 2} \begin{Bmatrix} g(4)[0] + f(3)[3] = 0 + 18 = 18 \\ g(4)[1] + f(3)[1] = 4 + 10 = 14 \\ g(4)[2] + f(3)[0] = 10 + 0 = 10 \end{Bmatrix} \rightarrow f(4) = 18, x(4) = 0$$

Za $Q = 3$:

$$f(4)[\ 3\ ] = \max_{0 \le x(4) \le 3} \begin{Bmatrix} g(4)[0] + f(3)[3] = 0 + 25 = 25 \\ g(4)[1] + f(3)[2] = 4 + 18 = 22 \\ g(4)[2] + f(3)[1] = 10 + 10 = 20 \\ g(4)[3] + f(3)[0] = 18 + 0 = 18 \end{Bmatrix} \rightarrow f(4) = 25, x(4) = 0$$

Za $Q = 4$:

$$f(4)[\ 4\ ] = \max_{0 \le x(4) \le 4} \begin{Bmatrix} g(4)[0] + f(3)[4] = 0 + 33 = 33 \\ g(4)[1] + f(3)[3] = 4 + 25 = 29 \\ g(4)[2] + f(3)[2] = 10 + 18 = 28 \\ g(4)[3] + f(3)[1] = 18 + 10 = 28 \\ g(4)[4] + f(3)[0] = 30 + 0 = 30 \end{Bmatrix} \rightarrow f(4) = 33, x(4) = 0$$

Za $Q = 5$:

$$f(4)[\ 5\ ] = \max_{0 \le x(4) \le 5} \begin{Bmatrix} g(4)[0] + f(3)[5] = 0 + 43 = 43 \\ g(4)[1] + f(3)[4] = 4 + 33 = 37 \\ g(4)[2] + f(3)[3] = 10 + 25 = 35 \\ g(4)[3] + f(3)[2] = 18 + 18 = 36 \\ g(4)[4] + f(3)[1] = 30 + 10 = 40 \\ g(4)[5] + f(3)[0] = 35 + 0 = 35 \end{Bmatrix} \rightarrow f(4) = 43, x(4) = 0$$

Za $Q = 6$:

$$f(4)[\,6\,] = \max_{0 \le x(4) \le 6} \begin{Bmatrix} g(4)[0] + f(3)[6] = 0 + 51 = 51 \\ g(4)[1] + f(3)[5] = 4 + 43 = 47 \\ g(4)[2] + f(3)[4] = 10 + 33 = 43 \\ g(4)[3] + f(3)[3] = 18 + 25 = 43 \\ g(4)[4] + f(3)[2] = 30 + 18 = 48 \\ g(4)[5] + f(3)[1] = 35 + 10 = 45 \\ g(4)[6] + f(3)[0] = 43 + 0 = 43 \end{Bmatrix} \rightarrow f(4) = 51, x(4) = 0$$

Za $Q = 7$:

$$f(4)[\,7\,] = \max_{0 \le x(4) \le 7} \begin{Bmatrix} g(4)[0] + f(3)[7] = 0 + 58 = 58 \\ g(4)[1] + f(3)[6] = 4 + 51 = 55 \\ g(4)[2] + f(3)[5] = 10 + 43 = 53 \\ g(4)[3] + f(3)[4] = 18 + 33 = 51 \\ g(4)[4] + f(3)[3] = 30 + 25 = 55 \\ g(4)[5] + f(3)[2] = 35 + 18 = 53 \\ g(4)[6] + f(3)[1] = 43 + 10 = 53 \\ g(4)[7] + f(3)[0] = 49 + 0 = 49 \end{Bmatrix} \rightarrow f(4) = 58, x(4) = 0$$

Za $Q = 8$:

$$f(4)[\,8\,] = \max_{0 \le x(4) \le 8} \begin{Bmatrix} g(4)[0] + f(3)[8] = 0 + 66 = 66 \\ g(4)[1] + f(3)[7] = 4 + 61 = 65 \\ g(4)[2] + f(3)[6] = 10 + 51 = 61 \\ g(4)[3] + f(3)[5] = 18 + 43 = 61 \\ g(4)[4] + f(3)[4] = 30 + 33 = 63 \\ g(4)[5] + f(3)[3] = 35 + 25 = 60 \\ g(4)[6] + f(3)[2] = 43 + 18 = 61 \\ g(4)[7] + f(3)[1] = 49 + 10 = 59 \\ g(4)[8] + f(3)[0] = 55 + 0 = 55 \end{Bmatrix} \rightarrow f(4) = 66, x(4) = 0$$

Za $Q = 9$:

$$f(4)[\,9\,] = \max_{0 \le x(4) \le 9} \begin{Bmatrix} g(4)[0] + f(3)[9] = 0 + 74 = 74 \\ g(4)[1] + f(3)[8] = 4 + 66 = 70 \\ g(4)[2] + f(3)[7] = 10 + 61 = 71 \\ g(4)[3] + f(3)[6] = 18 + 51 = 69 \\ g(4)[4] + f(3)[5] = 30 + 43 = 73 \\ g(4)[5] + f(3)[4] = 35 + 33 = 68 \\ g(4)[6] + f(3)[3] = 43 + 25 = 68 \\ g(4)[7] + f(3)[2] = 49 + 18 = 67 \\ g(4)[8] + f(3)[1] = 55 + 10 = 65 \\ g(4)[9] + f(3)[0] = 60 + 0 = 60 \end{Bmatrix} \rightarrow f(4) = 74, x(4) = 0$$

Za $Q = 10$:

$$f(4)[\,10\,] = \max_{0 \le x(4) \le 10} \begin{Bmatrix} g(4)[0] + f(3)[10] = 0 + 80 = 80 \\ g(4)[1] + f(3)[9] = 4 + 74 = 78 \\ g(4)[2] + f(3)[8] = 10 + 66 = 76 \\ g(4)[3] + f(3)[7] = 18 + 61 = 79 \\ g(4)[4] + f(3)[6] = 30 + 51 = 81 \\ g(4)[5] + f(3)[5] = 35 + 43 = 77 \\ g(4)[6] + f(3)[4] = 43 + 33 = 76 \\ g(4)[7] + f(3)[3] = 49 + 25 = 74 \\ g(4)[8] + f(3)[2] = 55 + 18 = 73 \\ g(4)[9] + f(3)[1] = 60 + 10 = 70 \\ g(4)[10] + f(3)[0] = 64 + 0 = 64 \end{Bmatrix} \rightarrow \begin{matrix} f(4) = 81 \\ x(4) = 4 \end{matrix}$$

Dobivene vrijednosti zatim uvrštavamo u tablicu rješenja:

Tablica 8. Rješenje problema investiranja 4/4 - vlastiti primjer

Q	$X(1)$	$f(1)[\,Q\,]$	$X(2)$	$f(2)[\,Q\,]$	$x(3)$	$f(3)[\,Q\,]$	$x(4)$	$f(4)[\,Q\,]$
0	0	0	0	0	0	0	0	0
1	1	10	0	10	0	10	0	10
2	2	17	1	18	0	18	0	18
3	3	25	0, 1	25	0	25	0	25
4	4	33	0, 1	33	0, 3, 4	33	0	33
5	5	36	1	41	4	43	0	43
6	6	41	2, 3	47	4	51	0	51
7	7	50	3	55	4	58	0	58
8	8	53	4	61	4	66	0	66
9	9	58	5	65	4	74	0	74
10	10	66	3	72	4	80	4	81

(Izvor: Vlastiti rad, 2021)

Iščitavanje dobivenih vrijednosti:

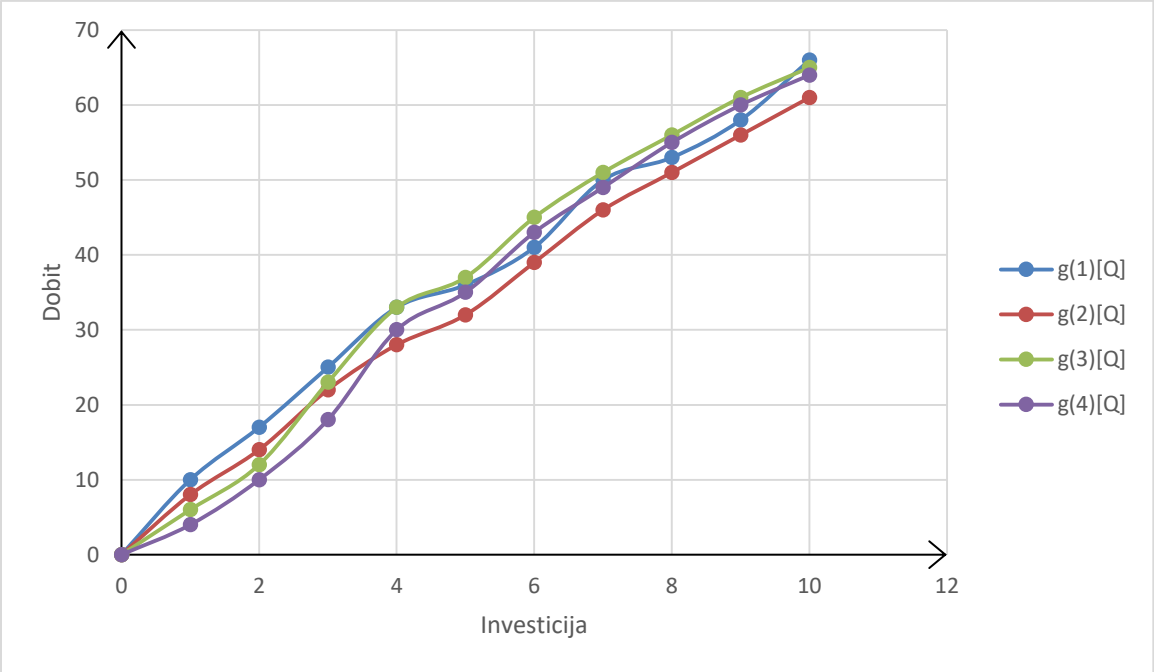
- **Pronalaženje optimalne odluke**, odnosno maksimalnog neto prihoda:
    - Iz tablice 12. očito je da se **maksimalni prihod** nalazi u četvrtoj etapi za $Q = 10$, te on iznosi 81 mil. kuna
    - Raspored sredstava za četvrtu etapu je prema tome $x(4)$ za koji je $f(4) = 81$, odnosno $x(4) = 4$
    - Dakle, od ukupno deset rata, **četiri se uzimaju za četvrtu** etapu
- **Preostalih šest rata** će se rasporediti na troetapni proces tako da:
    - U trećoj etapi tražimo maksimum te vidimo da je za $f(3) = 80$, pripadajući $x(3) = 4$, dakle **četiri rate pridružujemo trećoj etapi**
- **Preostale dvije rate** će se rasporediti na dvoetapni proces tako da:
    - U drugoj etapi prolazimo da je za $f(2) = 18$, pripadajući $x(2) = 1$, dakle **jednu ratu pridružujemo drugoj etapi**
    - Preostala rata se pridružuje prvoj etapi gdje je za $f(1) = 10$, pripadajući $x(1) = 1$, dakle **jednu ratu pridružujemo i prvoj etapi**

Dakle, za rezime rješenje je:

- **Maksimalan prihod** $f(4)[\,10\,] = 81$
- **Količina resursa** $Q = 10$
- **Količina resursa** raspoređena na **etapu** broj **jedan** je $x(1) = 1$
    - **Funkcija neto prihoda** za navedenu količinu resursa je $g(1)[\,1\,] = 10$
- **Količina resursa** raspoređena na **etapu** broj **dva** je $x(2) = 1$
    - **Funkcija neto prihoda** za navedenu količinu resursa je $g(2)[\,1\,] = 8$
- **Količina resursa** raspoređena na **etapu** broj **tri** je $x(3) = 4$
    - **Funkcija neto prihoda** za navedenu količinu resursa je $g(3)[\,4\,] = 33$
- **Količina resursa** raspoređena na **etapu** broj **četiri** je $x(4) = 4$
    - **Funkcija neto prihoda** za navedenu količinu resursa je $g(4)[\,4\,] = 30$

7.1.3. Grafički prikaz rješenja problema

Za dani primjer složene razdiobe ulaganja, radi lakše vizualizacije kretanja dobiti ovisno o ulaganju prilažem i sljedeći grafički prikaz.



Slika 15. Ovisnost dobiti o investicijama, vlastiti primjer (Vlastiti rad, 2021.)

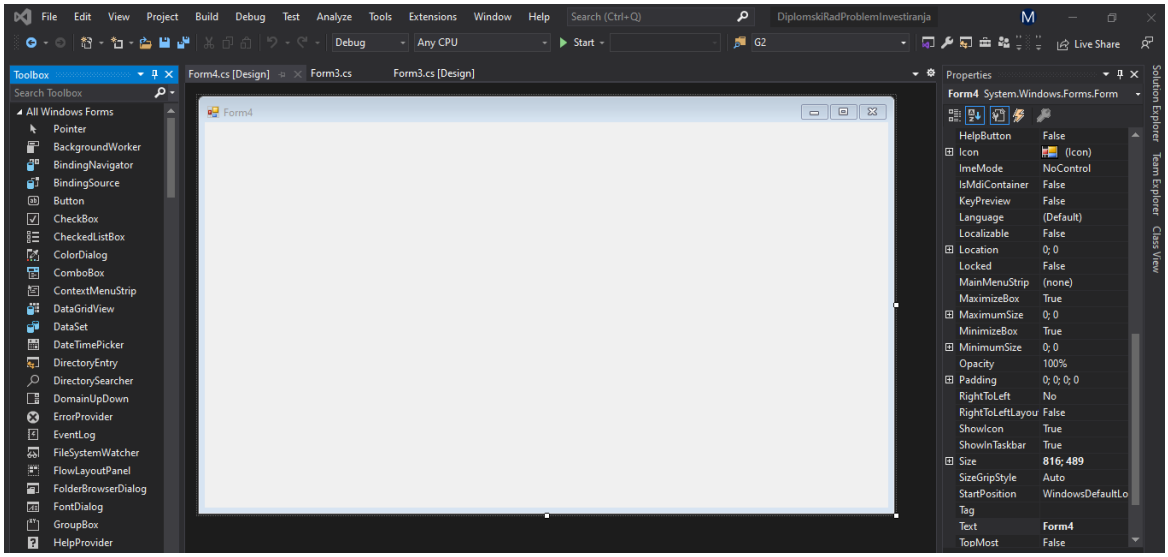
8. Programsko rješenje

Programsko rješenje za problem investiranja koristeći metodu dinamičkog programiranja, i navedeni primjer, implementirano je koristeći programski jezik C#.

Razvojno okruženje u kojem je riješen problem naziva se Visual Studio. To je okruženje koje je razvio Microsoft i koristi se ponajviše za razvijanje aplikacija koje koriste .NET okvire. Jedan od najkorištenijih programskih jezika u navedenom okruženju je upravo C#, ali koriste se i razni drugi, poput C++, Java i drugi.

Ovo okruženje i programski jezik koristilo se u izradi programskog rješenja jer pružaju potrebnu podlogu i pristup implementaciji koji odgovara rješavanju problema investiranja, koristeći metodu dinamičkog programianja. Navedeno se realiziralo uz pomoć Windows Forms-a. To je zapravo biblioteka koja pruža, kako arhitekturalnu, tako i vizualnu implementaciju. Omogućuje jednostavan „povuci i pusti“ način korištenja za dodavanje vizualnih komponenti. Vizualne komponente su primjerice gumb, tekstualno polje, naslovi polja...

Za implementaciju, prikaz što pojedina komponenta radi koriste se razni događaji, primjerice kada se klikne na gumb, otvori se novi prozor. Još jedan primjer je, kada se osvježi forma da se dohvate potrebne vrijednosti. Navedeni primjeri su samo mali dio onoga što Windows Forms pruža, a njegove mogućnosti bit će prikazane u poglavljima koja slijede prilikom opisa izrade programskog rješenja. Na sljedećoj slici prikazan je dizajnerski izgled Windows Forms.



Slika 16. Izgled Windows Forms (Vlastiti rad, 2021.)

8.1. Prikaz rada aplikacije

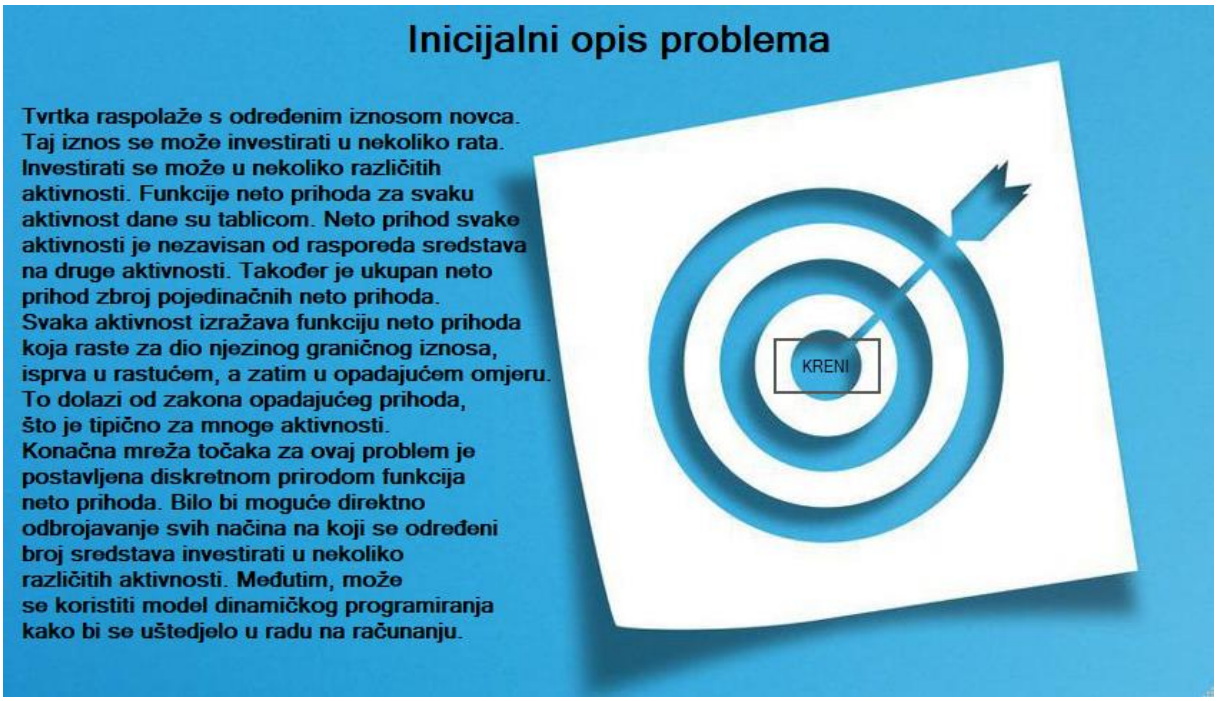
Nakon što je objašnjeno koji programski jezik, i programsko okruženje se koriste, predstoji predstaviti rad aplikacije.

Pokretanjem aplikacije, prva forma koja se prikazuje je forma sa naslovom diplomskog rada i autorom. Navedeno je prikazano na slici 17.



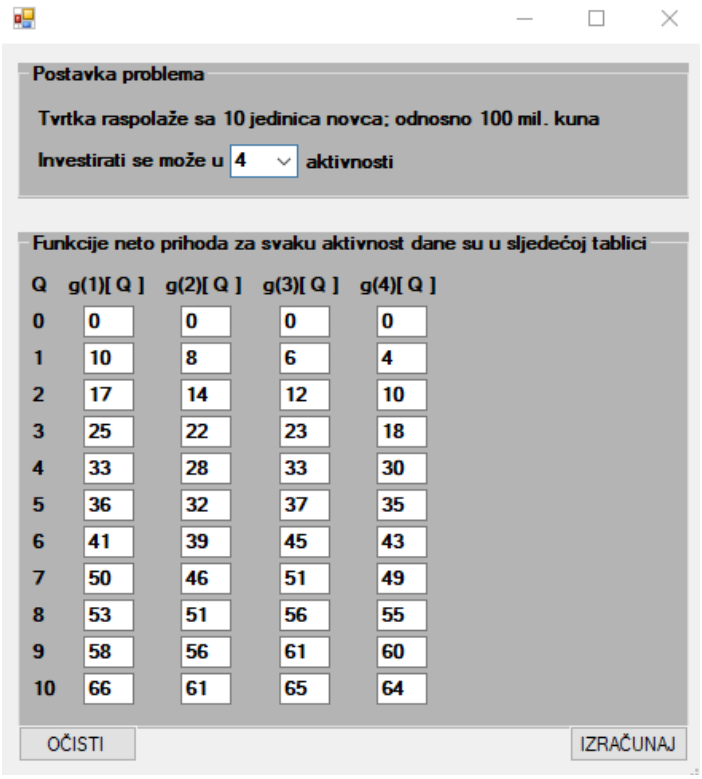
Slika 17. Početna forma u aplikaciji (Vlastiti rad, 2021.)

Na slici je vidljivo i da se na vrhu grafa kretanja troškova nalazi transparentni gumb. Klikom na gumb aplikacija kreće sa radom i otvara se forma sa opisom zadatka, koji je prikazan i u ručnom rješavanju problema. Opis problema je vidljiv na slici 18.



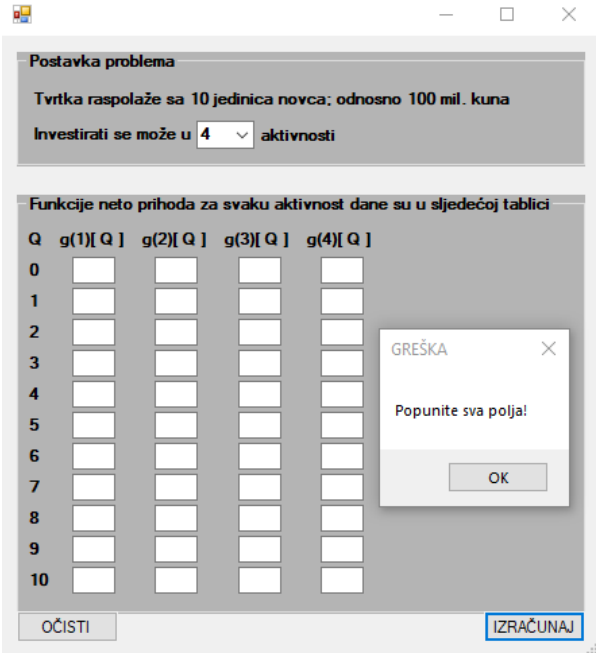
Slika 18. Forma s opisom problema (Vlastiti rad, 2021.)

Na slici 18., uz opis problema, vidljiv je i gumb sa tekstom kreni. Klikom na gumb otvara se forma za unos podataka i rješavanje problema investiranja, kako je vidljivo na slici 19. Dakle, odabrano je da se investirati može u četiri aktivnosti, dok su opcije još i tri, pet i šest aktivnosti.



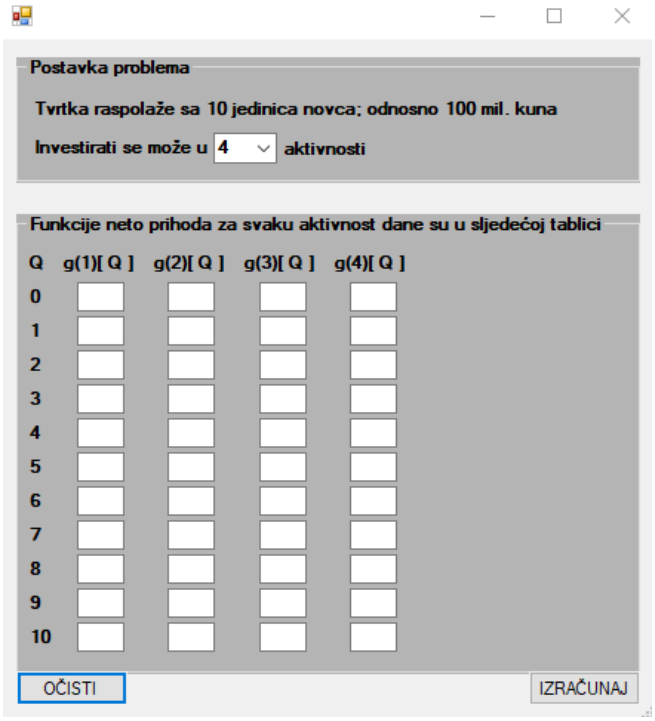
Slika 19. Forma za unos početnih podataka (Vlastiti rad, 2021.)

Na slici 19. vidljiva je inicijalna postavka problema, te su unesene sve vrijednosti kao što je to bio slučaj u računski riješenom primjeru. Klikom na gumb izračunaj pokreće se računanje potrebnih podataka. Ukoliko nije uneseno jedno polje ili više njih, javlja se greška, odnosno potrebno je unjeti sve podatke. Navedeno je vidljivo i na slici 20.



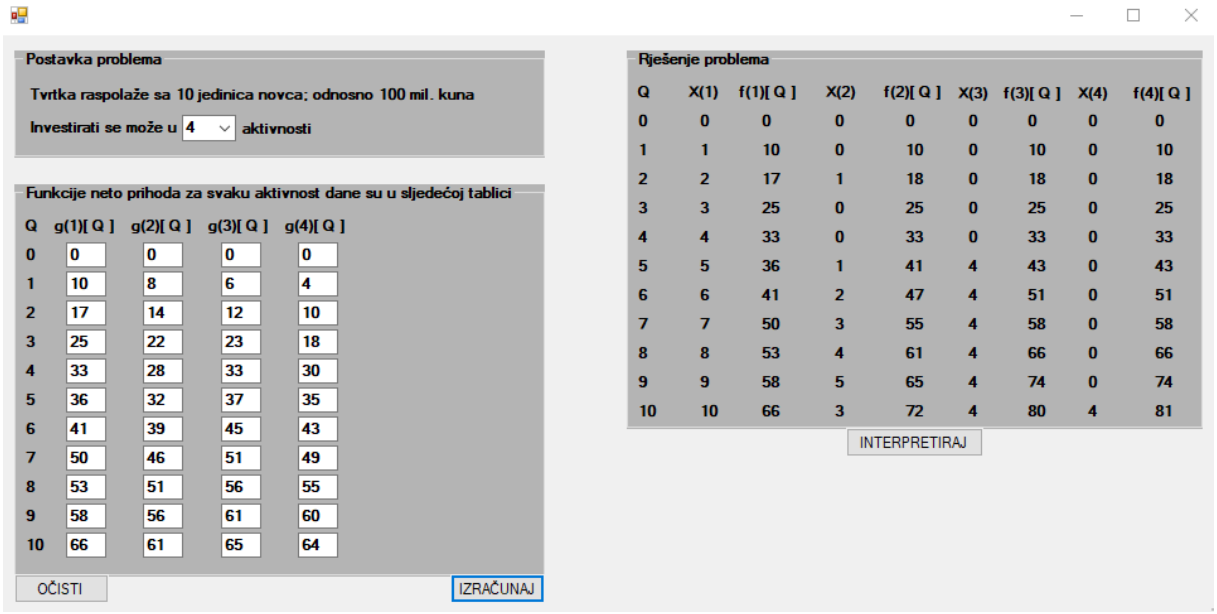
Slika 20. Greška – neispravan unos podataka (Vlastiti rad, 2021.)

Također, na formi se nalazi i gumb očisti. Klikom na navedeni gumb brišu se ulazni podaci, kako je vidljivo na slici 21.



Slika 21. Forma s obrisanim podacima (Vlastiti rad, 2021.)

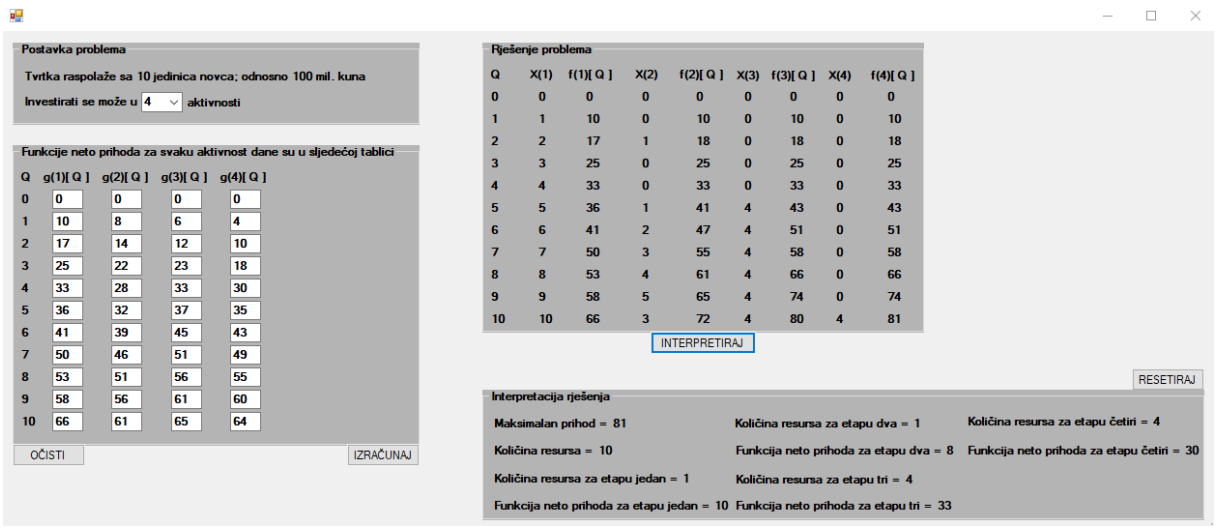
Ukoliko su sva polja unesena, aplikacija nastavlja s radom te se prikazuje tablično rješenje problema investiranja, kao na slici 22.



Slika 22. Forma s izračunatim podacima (Vlastiti rad, 2021.)

Usporedimo li dobiveno rješenje sa računskim rješenjem vidimo da je ono isto, uz izuzetak da se količina resursa (X) prikazuje samo za prvi pronađeni slučaj. Navedeno je ispravno jer, kako je već ranije navedeno, može se uzeti proizvoljna količina resursa bez utjecaja na daljnje rješenje problema.

Na slici 21. vidljiv je i gumb s tekstom interpretiraj, klikom na navedeni gumb prikazuje se interpretacija rješenja problema, kao što je vidljivo na slici 23.

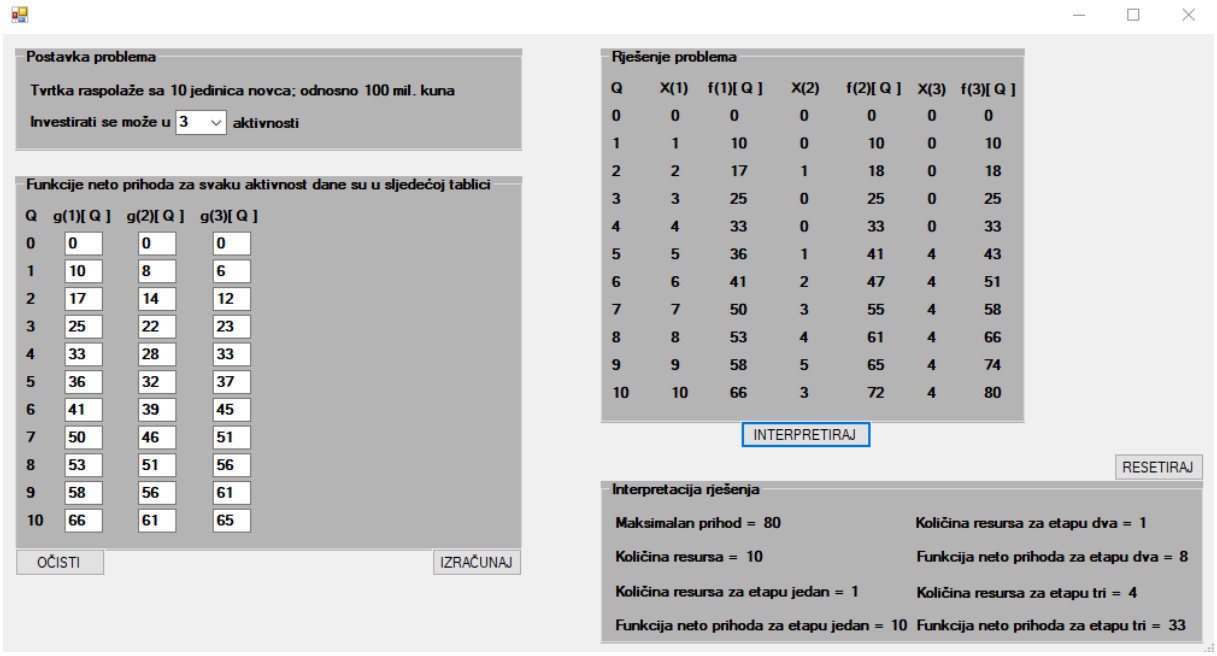


Slika 23. Forma s interpretiranim podacima (Vlastiti rad, 2021.)

Usporedimo li dobivene interpretirane podatke s onima koji su dobiveni računskim putem, vidimo da su oni u potpunosti isti, čime se potvrđuje ispravnost dobivenih podataka, te je problem investiranja ispravno rješen, kako ručno, tako i programski.

Na slici 23. vidljiv je i gumb s tekstom resetiraj, klikom na navedeni gumb forma se resetira, te je omogućen unos novih podataka i njihov izračun, kao što je bilo na slici 19.

Popuni li se forma s nekim novim podacima, postupak rješavanja će biti isti, ali naravno, rješenje je drukčije. Rješenje problema investiranja s novim ulaznim podacima prikazano je na slici 24. Na slici treba primjetiti da je odabrano investiranje u tri različite aktivnosti.

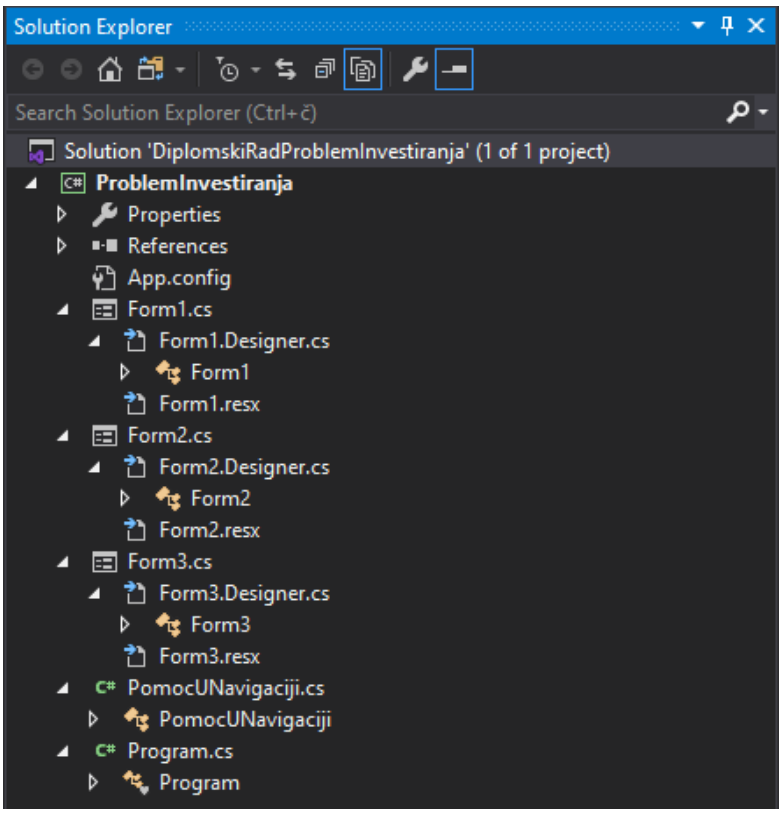


Slika 24. Forma rješenja s novim ulaznim podacima (Vlastiti rad, 2021.)

Nakon prikaza rada aplikacije, može se pristupiti prikazu implementacije problema, koja je prikazana u sljedećem potpoglavlju.

8.2. Implementacija

Implementacija programskog rješenja je kreirana koristeći programski jezik C# i programsko okruženje Visual Studio, kako je već i rečeno. Struktura programskog rješenja prikazana je na sljedećoj slici.



Slika 25. Struktura programskog rješenja (Vlastiti rad, 2021.)

Sa slike 25. je očito da se programsko rješenje sastoji od tri forme koje svaka imaju svoj dizajn i programski kod, od klase „Program.cs“ koja služi za pokretanje aplikacije, te od pomoćne klase, „PomocUNavigaciji.cs“, koja služi za lakše izmjenjivanje formi.

U sljedećem dijelu rada bit će analiziran programski kod svake od navedenih komponenti programskog rješenja.

8.2.1. Klasa Program.cs

Kao što je već rečeno, ova klasa služi za inicijalno pokretanje aplikacije, odnosno u ovom slučaju, prikazivanje početne forme aplikacije.

Programski kod:

```
static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
```

Većina programskog koda je zapravo automatski generirana od strane Visual Studia i služi za osnovnu postavku i mogućnost rada s Windows Formsima. Jedina izmjena s kojom korisnik radi je da definira što se događa nakon pokretanja aplikacije, odnosno nakon pozivanja metode `Application.Run()`. U ovom slučaju, očito je da se poziva otvaranje početne forme aplikacije.

8.2.2. Pomoćna klasa PomocUNavigaciji.cs

Kao što je već rečeno, ova klasa se koristi za lakše izmjenjivanje formi u aplikaciji, a njezin programski kod je:

```
public static class PomocUNavigaciji
    {
        public static void IdiNaFormu(Form trenutna, Form sljedeca)
        {
            trenutna.Hide();
            sljedeca.ShowDialog();
            trenutna.Close();
        }
    }
```

Očito je da se pomoćna klasa sastoji od samo jedne metode, `IdiNaFormu()`. Ova metoda se poziva kroz aplikaciju da se izbjegne ponavljanje koda, kako će biti vidljivo u sljedećem dijelu rada, a služi za otvaranje iduće forme, te zatvaranje trenutne.

8.2.3. Forma Form1

Početna forma aplikacije koja je više vizualne prirode i služi za prikaz naziva i autora projekta. Sastoji se samo od definiranja što se događa kada se klikne na gumb:

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Form2 form2 = new Form2();
        PomocUNavigaciji.IdiNaFormu(this, form2);
    }
}
```

Dakle, konstruktor forme služi za inicijaliziranje forme i dohvaćanje potrebnih komponenti kada se aplikacija pokrene. Klikom na gumb pokreće se događaj koji služi za prijelaz na sljedeću formu. Ovdje je iskorištena pomoćna klasa i njena već navedena metoda `IdiNaFormu()`.

8.2.4. Forma Form2

Slično kao i forma jedan, i ova forma je više vizualna bez puno logike u pozadini, a služi za opis problema investiranja. Također se sastoji od samo definiranja što se događa kada se klikne na gumb:

```
public partial class Form2 : Form
    {
        public Form2()
        {
```

```
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form3 form3 = new Form3();
            PomocUNavigaciji.IdiNaFormu(this, form3);
        }
    }
```

Kao i kod forme jedan, vidljiv je konstruktor koji se koristi za inicijaliziranje komponenti, te događaj kada se klikne na gumb. I ovdje je, također, iskorištena pomoćna klasa i njena metoda za prijelaz na sljedeću formu.

8.2.5. Forma Form3

Forma na koju je povezana zapravo cijela logika rješavanja problema investiranja. Sastoji se od mnogobrojnih metoda i događaja, a svaki od njih će biti prikazan kroz sljedeći dio rada.

8.2.5.1. Konstruktor i varijable

Korištene javne varijable i definiranje konstruktora je kako slijedi:

```
        private GroupBox[] groups;
        private bool flag;
        public Form3()
        {
            InitializeComponent();
            Size = new System.Drawing.Size(460, 500);
        }
```

Varijable koje su deklarirane koriste se radi dohvata stanja u tekstualnim poljima, odnosno radi provjera da li sva tekstualna polja koja se nalaze unutar grupa sadrže određenu vrijednost. U konstruktoru se odvija inicijaliziranje komponenti, te se postavlja početna visina i širina forme.

8.2.5.2. Metode

Metode koje forma koristi, koje će se kasnije pozivati u događajima, su kako slijedi:

- **`IzracunajLblFunkcija4()`** – metoda koja se koristi za izračunavanje vrijednosti funckije četiri, računa se na način da provjeri vrijednost količine resursa četiri, te nakon toga u tablici s ulaznim vrijednostima pronađe prihod za vrijednost količine resursa:

```
private string IzracunajLblFunkcija4()
  {
      if (int.Parse(lblKolResursa4.Text) == 1)
      {
          return txtG41.Text;
      }
      else if (int.Parse(lblKolResursa4.Text) == 2)
      {
          return txtG42.Text;
      }
      else if (int.Parse(lblKolResursa4.Text) == 3)
      {
          return txtG43.Text;
      }
      else if (int.Parse(lblKolResursa4.Text) == 4)
      {
          return txtG44.Text;
      }
      else if (int.Parse(lblKolResursa4.Text) == 5)
      {
          return txtG45.Text;
      }
      else if (int.Parse(lblKolResursa4.Text) == 6)
      {
          return txtG46.Text;
      }
      else if (int.Parse(lblKolResursa4.Text) == 7)
      {
          return txtG47.Text;
      }
      else if (int.Parse(lblKolResursa4.Text) == 8)
```

```
        {
            return txtG48.Text;
        }
        else if (int.Parse(lblKolResursa4.Text) == 9)
        {
            return txtG49.Text;
        }
        else if (int.Parse(lblKolResursa4.Text) == 10)
        {
            return txtG410.Text;
        }
        else
        {
            return "GREŠKA";
        }
    }
```

- **IzracunajLblFunkcija3()** – metoda koja se koristi za izračunavanje vrijednosti funckije tri, računa se na način da provjeri vrijednost količine resursa tri, te nakon toga u tablici s ulaznim vrijednostima pronađe prihod za vrijednost količine resursa. Programski kod je kao u metodi iznad.
- **IzracunajLblFunkcija2()** – metoda koja se koristi za izračunavanje vrijednosti funckije dva, računa se na način da provjeri vrijednost količine resursa dva, te nakon toga u tablici s ulaznim vrijednostima pronađe prihod za vrijednost količine resursa. Programski kod je kao u metodi iznad.
- **IzracunajLblFunkcija1()** – metoda koja se koristi za izračunavanje vrijednosti funckije jedan, računa se na način da provjeri vrijednost količine resursa jedan, te nakon toga u tablici s ulaznim vrijednostima pronađe prihod za vrijednost količine resursa. Programski kod je kao u metodi iznad.
- **IzracunajLblFunkcija5()** – metoda koja se koristi za izračunavanje vrijednosti funckije pet, računa se na način da provjeri vrijednost količine resursa pet, te nakon toga u tablici s ulaznim vrijednostima pronađe prihod za vrijednost količine resursa. Programski kod je kao u metodi iznad.
- **IzracunajLblFunkcija6()** – metoda koja se koristi za izračunavanje vrijednosti funckije šest, računa se na način da provjeri vrijednost količine resursa šest, te nakon toga u tablici s ulaznim vrijednostima pronađe prihod za vrijednost količine resursa. Programski kod je kao u metodi iznad.

- **`IzracunajLblKolResursa2i1()`** – metoda koja se koristi za izračunavanje vrijednosti količine resursa jedan i dva:

```
private string IzracunajLblKolResursa2i1()
  {
      int ukupnaKolicinaResursa = int.Parse(this.lblKolResursa.Text);
      int zbrojEtapa3i4 = int.Parse(lblKolResursa4.Text) +
      int.Parse(lblKolResursa3.Text);
      int izracun = (ukupnaKolicinaResursa - zbrojEtapa3i4) / 2;
      return izracun.ToString();
  }
```

Dakle, vrijednost se računa tako da se uzme ukupna količina resursa od koje se zatim oduzme zbroj količine resursa tri i četiri. Za dobivanje rješenja se rezultat podijeli s dva, budući da se radi o dvoetapnom procesu, te se vraća dobivena vrijednost.

- **`IzracunajVrijednostiX1()`** – metoda koja se koristi za izračunavanje količine resursa jedan:

```
private void IzracunajVrijednostiX1()
  {
      lblX10.Text = lblQ0.Text;
      lblX11.Text = lblQ1.Text;
      lblX12.Text = lblQ2.Text;
      lblX13.Text = lblQ3.Text;
      lblX14.Text = lblQ4.Text;
      lblX15.Text = lblQ5.Text;
      lblX16.Text = lblQ6.Text;
      lblX17.Text = lblQ7.Text;
      lblX18.Text = lblQ8.Text;
      lblX19.Text = lblQ9.Text;
      lblX110.Text = lblQ10.Text;
  }
```

Vrijednosti se izračunavaju tako da se uzmu iz tablice sa ulaznim podacima.

- **`IzracunajVrijednostiF1()`** – metoda koja se koristi za izračunavanje funkcije jedan:

```
private void IzracunajVrijednostiF1()
```

```
{
    lblF10.Text = txtG10.Text;
    lblF11.Text = txtG11.Text;
    lblF12.Text = txtG12.Text;
    lblF13.Text = txtG13.Text;
    lblF14.Text = txtG14.Text;
    lblF15.Text = txtG15.Text;
    lblF16.Text = txtG16.Text;
    lblF17.Text = txtG17.Text;
    lblF18.Text = txtG18.Text;
    lblF19.Text = txtG19.Text;
    lblF110.Text = txtG110.Text;
}
```

Vrijednosti se izračunavaju tako da se uzmu iz tablice sa ulaznim podacima.

- **`IzracunajVrijednostiX2F2()`** – metoda koja se koristi za izračunavanje vrijednosti količine resursa dva i funkcije dva. Metoda je podijeljena na jedanaest područja, a svako područje je naglašeno za koji Q se koristi. Za navedenu metodu prikazano je računanje za vrijednost područja jedan i dva:

```
#region za Q = 0
            lblX20.Text = lblQ0.Text;
            lblF20.Text = lblQ0.Text;
#endregion

#region za Q = 1
            int vrijednostG20F11 = (int.Parse(txtG20.Text) +
            int.Parse(lblF11.Text));
            int vrijednostG21F10 = (int.Parse(txtG21.Text) +
            int.Parse(lblF10.Text));
            lblF21.Text = Math.Max(vrijednostG20F11,
            vrijednostG21F10).ToString();
            if (int.Parse(lblF21.Text) == vrijednostG20F11)
            {
                lblX21.Text = "0";
            }
            else if (int.Parse(lblF21.Text) == vrijednostG21F10)
            {
                lblX21.Text = "1";
            }
```

```
#endregion
```

Dakle, za izračunavanje se traže maksimumi vrijednosti, kako je prikazano i u ručnom rješavanju zadatka, zatim se provjerava koja je vrijednost maksimalna i uzima se njena količina resursa. Za sve ostale vrijednosti Q postupak je isti.

- **`IzracunajVrijednostiX3F3()`** – metoda koja se koristi za izračunavanje vrijednosti količine resursa tri i funkcije tri. Metoda je podijeljena na jedanaest područja, a svako područje je naglašeno za koji Q se koristi. Programski kod je kao u metodi iznad.
- **`IzracunajVrijednostiX4F4()`** – metoda koja se koristi za izračunavanje vrijednosti količine resursa četiri i funkcije četiri. Metoda je podijeljena na jedanaest područja, a svako područje je naglašeno za koji Q se koristi. Programski kod je kao u metodi iznad.
- **`IzracunajVrijednostiX5F5()`** – metoda koja se koristi za izračunavanje vrijednosti količine resursa pet i funkcije pet. Metoda je podijeljena na jedanaest područja, a svako područje je naglašeno za koji Q se koristi. Programski kod je kao u metodi iznad.
- **`IzracunajVrijednostiX6F6()`** – metoda koja se koristi za izračunavanje vrijednosti količine resursa šest i funkcije šest. Metoda je podijeljena na jedanaest područja, a svako područje je naglašeno za koji Q se koristi. Programski kod je kao u metodi iznad.
- **`ClearTextboxes()`** – metoda koja se koristi za brisanje vrijednosti iz tekstualnih polja kad se klikne na gumb za resetiranje:

```
private void
ClearTextboxes(System.Windows.Forms.Control.ControlCollection
controls)
        {
            foreach (Control control in controls)
            {
                if (control is TextBox)
                {
                    if (((TextBox)control).Visible == true)
                    {
                        ((TextBox)control).Text = string.Empty;
                    }
```

```
                }
                ClearTextboxes(control.Controls);
            }
        }
```

Dakle, za svaku kontrolu na formi se provjerava kojeg je tipa, te ako je tekstualno polje i ako je vidljivo, onda se njegova vrijednost postavlja na prazan string.

- **`CheckTextboxes()`** – metoda koja se koristi za provjeru da li sva tekstualna polja imaju unešenu vrijednost kad se klikne na gumb za resetiranje forme:

```
private void CheckTextboxes()
  {
      flag = false;

      for (int i = 0; i < groups.Length && !flag; i++)
      {
            foreach (TextBox tb in
            groups[i].Controls.OfType<TextBox>())
          {
              if (tb.Text.Trim().Length == 0)
              {
                  flag = true;
              }
          }
      }

  }
```

Dakle, provjerava se svaka kontrola koja je tipa tekstualno polje, odnosno njena vrijedost. Ako nema vrijednost zastavica se postavi na istinitu te se kasnije može vrijednost zastavice dohvatiti i ispisati grešku.

- **`ZaBrojAktivnosti6()`** – metoda koja se koristi za prikazivanje potrebnih polja kada je izabran broj aktivnosti šest. Sastoji se od postavljanja vidljivosti određenih polja na vidljivo, a određenih na skriveno.
- **`ZaBrojAktivnosti5()`** – metoda koja se koristi za prikazivanje potrebnih polja kada je izabran broj aktivnosti pet. Sastoji se od postavljanja vidljivosti određenih polja na vidljivo, a određenih na skriveno.

- **`ZaBrojAktivnosti4()`** – metoda koja se koristi za prikazivanje potrebnih polja kada je izabran broj aktivnosti četiri. Sastoji se od postavljanja vidljivosti određenih polja na vidljivo, a određenih na skriveno.
- **`ZaBrojAktivnosti3()`** – metoda koja se koristi za prikazivanje potrebnih polja kada je izabran broj aktivnosti tri. Sastoji se od postavljanja vidljivosti određenih polja na vidljivo, a određenih na skriveno.

8.2.5.3. Događaji

Događaji koje forma koristi, odnosno komponente koje pozivaju događaje su kako slijedi:

- **`Form3_Load()`** – definira događaj kada se forma učita, odnosno definira što se događa kad se forma učita:

```
private void Form3_Load(object sender, System.EventArgs e)
{
    groups = new GroupBox[]
    {
        groupBox1, groupBox2
    };
    ZaBrojAktivnosti4();
}
```

Dakle, kada se forma učita, inicijaliziraju se grupe elemenata koje je kasnije potrebno dohvatiti i koristiti, kao na primjer za provjeru vrijednosti tekstualnih polja. Također, budući da je inicijalna vrijednost broja aktivnosti četiri, poziva se metoda `ZaBrojAktivnosti4()`.

- **`btnIzracunaj_Click()`** – definira događaj kada se klikne na gumb za izračunavanje problema investiranja:

```
private void btnIzracunaj_Click(object sender, System.EventArgs e)
  {
      CheckTextboxes();
      if (flag == true)
      {
          MessageBox.Show("Popunite sva polja!", "GREŠKA");
          return;
      }
```

```
        else
        {
            Size = new System.Drawing.Size(1260, 505);
            this.groupBoxRješenje.Visible = true;
            this.btnInterpretiraj.Visible = true;
            this.groupBoxInterpretacija.Visible = false;
            this.btnResetiraj.Visible = false;
            IzracunajVrijednostiX1();
            IzracunajVrijednostiF1();
            IzracunajVrijednostiX2F2();
            IzracunajVrijednostiX3F3();
            IzracunajVrijednostiX4F4();
        }
    }
```

Klikom na gumb za izračunavanje rješenja problema investiranja, najprije se vrši provjera jesu li sva potrebna polja popunjena, kroz već navedenu metodu `CheckTextboxes()`. Ukoliko sva polja nisu popunjena javit će se greška te će biti potrebno ponoviti unos. Ukoliko su sva polja popunjena, aplikacija nastavlja s radom te se postavljaju nova visina i širina forme da bi potrebni podaci bili vidljivi. Također, vidljivost komponenti koje je potrebno prikazati postavlja se na istinito. Za izračunati potrebne podatke, koriste se već navedene metode: `IzracunajVrijednostiX1()`,`IzracunajVrijednostiF1()`,`IzracunajVrijednostiX2F2()`,`IzracunajVrijednostiX3F3()`, te `IzracunajVrijednostiX4F4()`.Ukoliko je broj aktivnosti drugi broj, koriste se ostale potrebne metode.

- **`btnInterpretiraj_Click()`** – definira događaj kada se klikne na gumb za interpretiranje rješenja problema investiranja:

```
private void btnInterpretiraj_Click(object sender, System.EventArgs e)
  {
      Size = new System.Drawing.Size(1260, 570);
      this.groupBoxInterpretacija.Visible = true;
      this.btnResetiraj.Visible = true;
      this.lblMaxPrihod.Text = lblF410.Text;
      this.lblKolResursa.Text = lblQ10.Text;
      this.lblKolResursa4.Text = lblX410.Text;
      this.lblFunkcija4.Text = IzracunajLblFunkcija4();
      this.lblKolResursa3.Text = lblX310.Text;
      this.lblFunkcija3.Text = IzracunajLblFunkcija3();
      this.lblKolResursa2.Text = IzracunajLblKolResursa2i1();
```

```
        this.lblFunkcija2.Text = IzracunajLblFunkcija2();
        this.lblKolResursa1.Text = IzracunajLblKolResursa2i1();
        this.lblFunkcija1.Text = IzracunajLblFunkcija1();
    }
```

Klikom na gumb interpretiraj, najprije se postavljaju nova visina i širina forme da da bi se mogli vidjeti svi potrebni podaci. Također, vidljivost komponenti koje se žele prikazati postavlja se na istinito. Nadalje, za svaku pojedinu komponentu u interpretaciji koristi se određena metoda, ili druga komponenta, za dohvaćanje njene vrijednosti. Ukoliko je broj aktivnosti drugi broj, koriste se ostale potrebne metode.

- **`btnResetiraj_Click()`** – definira događaj kada se klikne na gumb za resetiranje forme:

```
private void btnResetiraj_Click(object sender, System.EventArgs e)
    {
        Size = new System.Drawing.Size(460, 500);
        this.groupBoxRješenje.Visible = false;
        this.btnInterpretiraj.Visible = false;
        this.groupBoxInterpretacija.Visible = false;
        this.btnResetiraj.Visible = false;
    }
```

Klikom na gumb resetiraj, smanjuju se visina i širina forme na inicijalnu, te se sakrivaju komponente koje više ne trebaju biti vidljive.

- **`cmbOdabirBrojaAktivnosti_SelectedIndexChanged()`** – definira događaj kada se odabere broj aktivnosti:

```
private void cmbOdabirBrojaAktivnosti_SelectedIndexChanged(object
sender, EventArgs e)
        {
            string selectedItem =
cmbOdabirBrojaAktivnosti.Items[cmbOdabirBrojaAktivnosti.SelectedIndex
].ToString();
            if (int.Parse(selectedItem) == 6)
            {
                ZaBrojAktivnosti6();
            }
```

```
            else if (int.Parse(selectedItem) == 5)
            {
                ZaBrojAktivnosti5();
            }
            else if (int.Parse(selectedItem) == 4)
            {
                ZaBrojAktivnosti4();
            }
            else if (int.Parse(selectedItem) == 3)
            {
                ZaBrojAktivnosti3();
            }
        }
```

Odabirom određenog broja aktivnosti, poziva se potrebna metoda za izračunavanje problema za odabrani broj aktivnosti.

- **`btnOcisti_Click()`** – definira događaj kada se klikne na gumb za brisanje ulaznih podataka forme:

```
private void btnOcisti_Click(object sender, EventArgs e)
  {
      ClearTextboxes(this.Controls);
  }
```

Klikom na gumb očisti, brišu se uneseni ulazni podaci.

9. Zaključak

Proces na kojem se zasniva metoda dinamičkog programiranja, odnosno rješavanje po fazama gdje se u proračunu svake faze koriste optimalne vrijednosti dobivene u prethodnoj fazi naziva se Bellmanov princip. Rezultat navedenog principa je slijed optimalnih odluka koje se uspješno primjenjuju kod niza problema. Navedeno ima široku primjenu u gotovo svim segmentima operacijskih istraživanja, od linearnog i cjelobrojnog programiranja, preko analiza mreža do problema skladištenja i slično.

Svoju primjenu u navedenom području pronalazi i problem investiranja. Za problem investiranja se može reći da je to tipičan primjer problema dinamičkog programiranja koji ne sadrži vremensku komponentu. No, kod njega se može umjetno izvršiti dekompozicija na više etapa, što odgovara nizu aktivnosti. Nakon obavljene dekompozicije može se primijeniti načelo optimizacije metodom dinamičkog programiranja.

Primjer definiran u ovom radu prikazuje problem investiranja, a osmišljen je od strane autora rada na temelju primjera profesora Dobrenića, riješen je kako računski, tako i programski. Programsko rješenje zadatka potvrdilo je ispravnost ručno dobivenih podataka. Programsko rješenje je poslužilo i pri uklanjanju pogrešaka u ručnom rješenju problema.

Prikazani primjer problema investiranja je samo jedan od mnogobrojnih mogućih slučajeva i načina kako pristupiti rješavanju navedenog. Najprimjenjeniji način pristupa je upravo koristeći metodu dinamičkog programiranja jer on smanjuje mogućnost pogrešaka i značajno smanjuje raspisivanje problema.

Pristup problemu investiranja i njegovo rješavanje metodom dinamičkog programiranja može biti korisno i u svakodnevnom životu, prilikom odlučivanja vezanog za investiranje u određene resurse. Postavi li se problem pravilno, te ukoliko se razmotri kretanje tržišta i uzme dovoljno vremena pri donošenju odluka, investitor može značajno popraviti svoje prihode.

Kao što je u samom poglavlju investiranja rečeno, najbitnije je biti strpljiv, proučiti tržište, te dočekati pravu priliku za ulaganje. Uzevši u obzir i metodu dinamičkog

programiranja, može se zaključiti da se dobrom analizom i računanjem mogu lako postići zacrtani ciljevi ulaganja.

Popis literature

1. Graham, B.; [prevoditelj Silvije Orsag]; (2006). *Inteligentni investitor*. Zagreb: Masmedia.

2. Picardo, E. (2021). *Investing.* Preuzeto 09.05.2021. s https://www.investopedia.com/terms/i/investing.asp

3. Thakur, M. (2021.). *Examples of Investment Types*. Preuzeto 09.05.2021. s https://www.wallstreetmojo.com/investment-examples/

4. *AMZN Historical Data*. (2021). Nasdaq. Preuzeto 09.05.2021. s https://www.nasdaq.com/market-activity/stocks/amzn/historical

5. Winston, W. L. (1994). *Operation Research Aplications and Algorithms*. SAD, California: Duxbury Press.

6. Petrić, J. J. (1979). *Operaciona istraživanja: Knjiga prva: Peto izdanje*. Beograd, Srbija: Savremena administracija.

7. Dobrenić, S. (1978). *Operativno istraživanje*. Varaždin: Fakultet organizacije i informatike.

8. Kalpić, D., i Mornar, V. (1996). *Operacijska istraživanja*. Zagreb: ZEUS.

9. Barković, D. (2001). *Operacijska istraživanja: Drugo izmjenjeno i dopunjeno izdanje.* Osijek: Ekonomski fakulter.

10. Andreani, P. (2018). *Chapter 11 Dynamic Programming*. Unicamp. Preuzeto 14.05.2021. s https://www.ime.unicamp.br/~andreani/MS515/capitulo7.pdf

11. Jacobs, O.L.R. (1967). *An Introduction to Dynamic Programming – The Theory of Multistage Decision Processes*. London, Engleska: Chapman and Hall Ltd.

12. Četirkin, E.M. (1971). *Teorija masovogo obsluženija i eje primenenie v ekonomike*. Rusija, Moska: Statistika.

13. Nemhauser, G.L. (1966). Introduction to Dynamic Programming. SAD, New York: John Wiley & Sons, Inc.

14. Fabricky, W.J., i Torgersen, P. (1966). *Operations Economy*. SAD, New Jersey: Prentice-Hall, Inc.

15. Gillett, B.E. (1976). *Introduction to Operations Research – Computer-oriented Algorithmic Approach*. SAD, New York: McGraw-Hill, Inc.

16. Kaufmann, A. (1967). *Methodes et modeles de la recherche operationnelle*. Francuska, Pariz: Dunod.

17. Richardson, L. (1996). *Investiranje: Dio I*. Zagreb: Biblioteka.

18. Bernstein, P.L., i Damodaran, A. (1998). *Investment Management*. SAD, New York: John Wiley & Sons, Inc.

19. Cvjetičanin, M. (2004). *Burzovno trgovanje: Priručnik za investitore i analitičare.* Zagreb: Masmedia.

20. Tyson, E. (2017). *Investing for dummies, 8th Edition*. SAD, New Jersey: John Wiley & Sons, Inc.

21. Ravindran, A., Phillips, D.T., Solberg, J.J. (2007). *Operation research: principles and practices*. India: John Wiley & Sons, Ltd.

22. Vukičević, M. (2000). *Financiranje malih poduzeća*. Zagreb: Sveučilišna tiskara.

Popis slika

Slika 1. Izgradnja modela (Ravindran, A. i suradnici, 2007) 3
Slika 2. Primjer načina rada dinamičkog programiranja (Podrijetlo izraza, 2020) 11
Slika 3. Struktura determinističkog dinamičkog programiranja (Andreani, P., 2018) 14
Slika 4. Struktura probabilističkog dinamičkog programiranja (Andreani, P., 2018) 15
Slika 5. Vremenski diskretna varijabla (Dobrenić, S., 1978, str. 324) 17
Slika 6. Vremenski kontinuirana varijabla (Dobrenić, S., 1978, str. 324) 18
Slika 7. Tipovi investiranja (Thakur, M., 2021.) 25
Slika 8. Podjela vrijednosnica (Cvjetičanin, M. , 2005, str. 77) 27
Slika 9. Podjela dionica (Cvjetičanin, M. , 2005, str. 78) 27
Slika 10. Podjela obveznica po mjestu izdavatelja (Cvjetičanin, M. , 2005, str. 99) 29
Slika 11. Podjela obveznica po tipu izdavatelja (Cvjetičanin, M. , 2005, str. 100) 29
Slika 12. Podjela investicijskih fondova po tipu (Cvjetičanin, M. , 2005, str. 127) 31
Slika 13. Prikaz dinamičkog programiranja pomoću mrežnog dijagrama (Fabricky, W.J., i Torgersen, P. , 1966, str. 423). 38
Slika 14. Ovisnost dobiti o investicijama (Kaufmann, 1970) 41
Slika 15. Ovisnost dobiti o investicijama, vlastiti primjer (Vlastiti rad, 2021.) 55
Slika 16. Izgled Windows Forms (Vlastiti rad, 2021.) 56
Slika 17. Početna forma u aplikaciji (Vlastiti rad, 2021.) 57
Slika 18. Forma s opisom problema (Vlastiti rad, 2021.) 58
Slika 19. Forma za unos početnih podataka (Vlastiti rad, 2021.) 58
Slika 20. Greška – neispravan unos podataka (Vlastiti rad, 2021.) 59
Slika 21. Forma s obrisanim podacima (Vlastiti rad, 2021.) 59
Slika 22. Forma s izračunatim podacima (Vlastiti rad, 2021.) 60
Slika 23. Forma s interpretiranim podacima (Vlastiti rad, 2021.) 60
Slika 24. Forma rješenja s novim ulaznim podacima (Vlastiti rad, 2021.) 61
Slika 25. Struktura programskog rješenja (Vlastiti rad, 2021.) 62

Slika 2. Preuzeto 13.05.2021. s https://hr.puntomarinero.com/introduction-to-dynamic-programming/

Slika 3. Preuzeto 14.05.2021. s https://www.ime.unicamp.br/~andreani/MS515/capitulo7.pdf

Slika 4. Preuzeto 14.05.2021. s https://www.ime.unicamp.br/~andreani/MS515/capitulo7.pdf

Slika 7. Preuzeto 09.05.2021. s https://www.wallstreetmojo.com/investment-examples/

Popis tablica

Tablica 1. Kretanje dionica tvrtke Amazon 34
Tablica 2. Prikaz sheme računanja 39
Tablica 3. Očekivana dobit za pojedina područja 40
Tablica 4. Funkcije neto prihoda – vlastiti primjer 43
Tablica 5. Rješenje problema investiranja 1/4 - vlastiti primjer 44
Tablica 6. Rješenje problema investiranja 2/4 - vlastiti primjer 47
Tablica 7. Rješenje problema investiranja 3/4 - vlastiti primjer 50
Tablica 8. Rješenje problema investiranja 4/4 - vlastiti primjer 53