Korištenje metode dinamičkog programiranja u rješavanju problema nabave

Krnetić, Bruno

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Bruno Krnetić

Korištenje metode dinamičkog programiranja u rješavanju problema nabave

DIPLOMSKI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Bruno Krnetić
JMBAG: 0016103890
Studij: Informacijsko i programsko inženjerstvo

Korištenje metode dinamičkog programiranja u rješavanju problema nabave

DIPLOMSKI RAD

Mentor:
Doc. dr. sc. Nikolina Žajdela Hrustek

Varaždin, srpanj 2019.

Bruno Krnetić

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

Cilj izrade ovog diplomskog rada jest upoznati se sa mogućnostima korištenja algoritma dinamičkog programiranja za rješavanje problema nabave. Prije detaljnije razrade algoritma dinamičkog programiranja i primjene istog na primjerima, definirani su pojmovi zaliha i nabave, prikazani problemi te objašnjen značaj istih za poduzeće. Zatim je opisan problem nabave na realnom primjeru te je primijenjen algoritam dinamičkog programiranja na istome. U sklopu ovog diplomskog rada algoritam dinamičkog programiranja za rješavanje problema nabave implementiran je u obliku programskog rješenja objašnjenog i prikazanog na kraju rada. Minimizacija odnosno uklanjanje nepotrebnih troškova nastalih tijekom procesa nabave uvjetovani su pravovremenim planiranjem iste. Algoritam dinamičkog programiranja jedan je od alata koji olakšava planiranje nabave i omogućuje smanjenje troškova iste.

Ključne riječi: zalihe; nabava; planiranje nabave; dinamičko programiranje; problem nabave; višeetapni procesi; razdoblje; čuvanje zaliha; algoritam.

Sadržaj

1. Uvod

Tema ovog diplomskog rada jest Korištenje metode dinamičkog programiranja u rješavanju problema nabave. Rad je podijeljen na teorijski i praktični dio. Glavni dio teoretske razrade sadrži opis same metode i prikaz algoritma dinamičkog programiranja, dok se praktični dio odnosi na primjenu metode za rješenje problema nabave vode, odabranog poduzeća Q d.o.o. Praktični dio također uključuje i programsko rješenje realizirano u programskom jeziku *C#*. Korišteni podaci u praktičnoj razradi prikupljeni su intervjuiranjem zaposlenika u odabranom poduzeću te analizom podataka dobivenih u kartici partnera koristeći alat *MS Excel*. Rezultati primjene metode dinamičkog programiranja na praktičnom primjeru prikazani su tablično.

Za izradu rada primarno je korištena relevantna znanstvena i stručna literatura, dok ostatak čine relevantni materijali pronađeni na web stranicama navedenima u literaturi.

Predmet odnosno svrha ovog diplomskog rada jest usvojiti nova znanja na području operacijskih istraživanja, točnije rješavanja problema nabave. Cilj rada je istražiti i predočiti postupak korištenja metode dinamičkog programiranja za uspješnu optimizaciju procesa nabave.

Tijekom pisanja ovog diplomskog rada korištene su razne znanstvene metode poput metode analize i sinteze, metode klasifikacije, metode deskripcije, metode apstrakcije i konkretizacije, komparativna metoda, metoda intervjua te matematička metoda. Sam rad podijeljen je u 8 dijelova opisanih u nastavku.

Prvi dio jest uvod, u kojem su navedeni cilj i predmet rada, korištene znanstvene metode, način prikupljanja podataka za praktičnu razradu, kratak opis i obrazloženje strukture rada i korištene literature.

Prije razrade same metode dinamičkog programiranja, potrebno je razumijeti pojmove zaliha i nabave. Tako je u drugom dijelu teoretski obrađen i definiran pojam zaliha, navedene su i opisane vrste zaliha, značaj zaliha za poduzeće te problemi zaliha.

Treći dio obuhvaća teoretsku razradu i definiranje pojma nabave te značaja iste u poduzeću. Zatim je opisan sam proces nabave, navedeni su sustavi nabave te problemi istih.

Nakon definiranja zaliha i nabave, počinje razrada osnovnog dijela ovog diplomskog rada, točnije dinaičkog programiranja. Tako je u četvrtom dijelu objašnjen algoritam dinamičkog programiranja, definirani su osnovni pojmovi i označavanja, opisani su višeetapni procesi te značaj istih za dinamičko programiranje i prikazana je primjena metode dinamičkog programiranja kroz zadani primjer.

Peto poglavlje sadrži opis poduzeća Q d.o.o. Navedene su i objašnjene djelatnost kojom se bavi i usluge koje nudi.

U šestom poglavlju opisan je problem nabave na stvarnom primjeru spomenutom prije. Pri tome je opisan proces nabave vode u poduzeću te proces prikupljanja stvarnih podataka nad kojima je potom primijenjena metoda dinamičkog programiranja.

U sedmom poglavlju prikazano je programsko rješenje za rješavanje problema nabave korištenjem metode dinamičkog programiranja, realizirano u programskom jeziku *C#*. U priloženom primjeru korištenja aplikacije korišteni su stvarni podaci opisani u poglavlju prije.

Na kraju rada odnosno u osmom poglavlju izveden je zaključak u kojem su iznesena razmišljanja autora rada te rezimirani najvažniji rezultati i doprinosi razrade teme ovog rada.

2. Zalihe

Zalihe su neizostavan dio svakog poduzeća te su temelj procesa proizvodnje i prodaje, a time i nabave. Stoga velik dio ulaganja pojedinog poduzeća upravo odlazi na zalihe.

Pregledom relevantne literature iz područja vezanog uz zalihe utvrdilo se da postoji više definicija zaliha. Tako Majstorović[8] navodi kako se zalihama smatraju uskladišteni materijali koji se koriste s ciljem održanja kontinuirane proizvodnje i zadovoljnih kupaca. Nešto konkretnija definicija nalazi se u Ammerovoj *Dictionary of Business and Economics* [1], gdje pod zalihama podrazumijeva vlastiti materijal koji se koristi u poslovanju, a namijenjen je unutarnjoj potrošnji ili prodaji te kao takav uključuje sirovine, poluproizvode, materijal u radu i gotove proizvode.

Postoji još nekolicina definicija zaliha, no navedene dvije definicije objedinjuju ono što stoji u ostalima te kao takve najbolje odgovaraju kontekstu ovoga rada. Skraćenu verziju dvaju navedenih možemo svesti na to da zalihe možemo definirati kao količinu robe koja se nalazi u skladištu, a nije u neposrednoj uporabi nekog procesa, što je upravo i osnova za daljnju razradu ove teme.

2.1. Vrste zaliha

Prije opisa vezanih uz probleme zaliha bitno je iste rasporediti po vrstama, obzirom da su različite vrste karakteristične za rješenje odgovarajućeg problema. U objašnjenjima pojedine vrste kriju se i problemi koje određena vrsta rješava. Tako zalihe možemo klasificirati na dva načina [6]:

1. Prema fazama u kojima se nalaze tijekom proizvodnog procesa :
   - zalihe sirovina;
   - zalihe nedovršene proizvodnje (materijali uključeni u proizvodni proces);
   - zalihe gotovih proizvoda.
2. Prema planiranom stanju i potrebi za kontinuiranjem izvršavanjem procesa proizvodnje odnosno prodaje:
   - minimalne – najmanja količina potrebna za zadovoljenje obaveza poduzeća po količini i potrošnji;
   - maksimalne – gornja granica skladištene robe koja nalaže u kojem razdoblju se ne smije nabavljati roba;

- optimalne – količina robe dovoljna za redovnu i potpunu opskrbu proizvodnje ili kupca, a uz minimalne troškove skladištenja i nabavljanja;
- prosječne – prosječna količina robe za zadano razdoblje;
- sigurnosne – količina robe koja osigurava spremnost na neočekivane promjene u ponudi ili potražnji;
- špekulativne – količina robe koja se nabavi s ciljem prodaje kod povećanja cijene iste;
- sezonske – količina robe sačuvana za periode u kojima se očekuje veća potražnja za istom;
- nekonkurentne – količina skladištene robe koja je iz određenih razloga izgubila na vrijednosti te se teško može prodati.

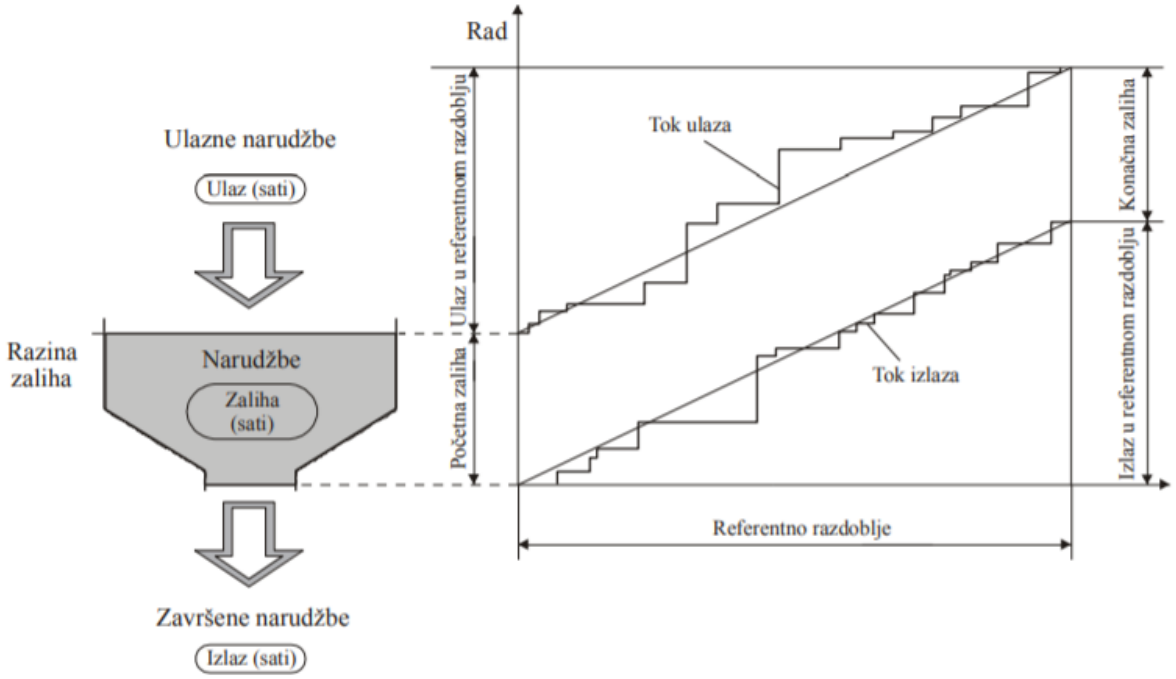
2.2. Problemi zaliha

Proces upravljanja zalihama može uvelike ovisiti o vanjskim faktorima koji mogu predstavljati razne prepreke određivanju optimalne politike istog. Neki od tih problema su [7]:

- nepredvidivost potražnje;
- nepouzdana nabava i isporuka (kašnjenje isporuke);
- varirajuća kvaliteta proizvoda;
- velik ili nedovoljan broj artikala;
- kratko trajanje potražnje za određeni proizvod.

Najčešće svi navedeni problemi proizlaze upravo iz prvog problema, nestabilne potražnje koja može varirati u nepravilnim razmacima. Pri tome, postoje dva modela potražnje: zavisni i nezavisni model. Nezavisni model prikazuje potražnju u nepravilnim vremenskim razmacima, a koja je uvjetovana od strane tržišta, pa se zalihe nadopunjuju izlaskom iz skladišta. S druge strane, zavisni model prikazuje potražnju pri kojoj se potrebna količina robe i razdoblje nabavljanja mogu predvidjeti.

Kako bismo mogli što bolje upravljati planiranjem zaliha, bitno je prikazati tijek kretanja zaliha kroz poduzeće. Jedan od usvojenih načina prikaza navedenog jest model lijevka (Slika 1.). U navedenom modelu u obzir se uzima promatrani segment poduzeća (radno mjesto, organizacijski odjel, cijelo poduzeće...) te transformacija promatranog predmeta (roba, radni nalozi, narudžbe...) kroz faze ulaza, zadržavanja te izlaza. Tako desni dio slike prikazuje tokove ulaza i izlaza za zadani referentni period [12].



Slika 1: Model lijevka kao prikaz zaliha (Izvor: Wiendahl, Load, 1995)

Za tok ulaza potrebno je odrediti količinu početnih zaliha pri početku promatranog razdoblja. Krivulja ulaznog toka počinje od ishodišta koordinatnog sustava te se bazira na sumi ulaznih naloga u određenom vremenu. Na isti način određen je i izlazni tok, no isti uključuje završene narudžbe u određenim vremenskim razmacima. Na kraju razdoblja dobivena razlika između ulaznih i izlaznih količina zaliha predstavlja konačnu zalihu. Cilj je da ista odgovara potrebnim količinama početnih zaliha u sljedećem razdoblju. Na taj način direktno se utječe na smanjenje nepotrebnih troškova nastalih nemogućnošću prodaje viška robe (kvarenje, propadanje, gubitak vrijednosti...) ili se uklanja rizik od primoranog prekida proizvodnje zbog manjka zaliha [12].

Zaključno, svrha kvalitetnog upravljanja zalihama može se svesti na nekoliko točaka [8]:

- pripremiti poslovanje na neočekivane promjene;
- osigurati ekonomičnu nabavu i proizvodnju;
- omogućiti kontinuirani nadzor nad zalihama;
- održati tok materijala unutar poslovnog sustava;
- spriječiti utjecaj promjena u ponudi i potražnji na poslovanje poduzeća.

3. Nabava

Uz proces proizvodnje i prodaje, proces nabave jedan je od ključnih procesa svakog poduzeća te kao takav igra bitnu ulogu u uspjehu istoga. Sam pojam nabave možemo definirati na dva načina, kao funkciju i kao djelatnost [3].

Tako nabava predstavlja poslovnu funkciju u smislu poslovne organizacijske jedinice čiji je osnovni zadatak nabavljati robu i potrebna dobra za proizvodnju i/ili prodaju. Radnje koje uključuje proces nabave skupno čine djelatnost nabave.

3.1. Nabava u užem i širem smislu

Uloga i proces nabave u poduzeću mogu se opisati kroz prikaz nabave u užem i širem smislu. U užem smislu nabava predstavlja skup svakodnevnih operativnih poslova čiji je osnovni cilj opskrbiti poduzeće objektima nabave u potrebnim količinama, po što povoljnijoj cijeni i što bolje kvalitete, a da se pri tom poštuju rokovi isporuke. Isti se mogu podijeliti na [4]:

- primanje, ispitivanje i objedinjavanje zahtjeva za nabavu;
- upiti dobavljačima;
- prijam i vrednovanje ponuda;
- vođenje pregovora;
- izbor dobavljača i naručivanje;
- praćenje rokova isporuke, prijam i ispitivanje naručene robe i pratećih dokumenata;
- reklamacije, kontrola zaliha, izvještavanje.

Objekti nabave u užem smislu mogu biti: sirovine, pogonski i pomoćni materijal, dijelovi, sklopovi, solucije raznih poslovnih rješenja i trgovačka roba [14].

U širem smislu nabava obuhvaća domenu strateški orjentiranih poslova neophodnih za ostvarenje željenog učinka i dobiti pojedinog poduzeća. Ti poslovi su [4]:

- stohastičko i determinističko istraživanje potreba;
- analiza i planiranje nabave (utvrđivanje potrebnih količina, analiza ukupnih troškova proizvoda, vrijednosna analiza, izračun ekonomičnih normativa zaliha);

- benchmarking nabave (usporedba procesa nabave s najboljim kompanijama s ciljem prilagodbe i primjene utvrđenih prednosti istih na vlastito poduzeće)
- izbor sustava nabavljanja i kontrola nabave;
- upravljanje mrežom opskrbe;
- istraživanje tržišta nabave;
- suradnja kod razvoja novih proizvoda;
- priprema i sklapanje dugoročnih ugovora o nabavi;
- kooperacija u nabavi;
- upravljanje ljudskim potencijalom u nabavi.

Osnovni objekti nabave u širem smislu su, pored prethodno spomenutih, i usluge (čišćenje, održavanje, konzultantske usluge, istraživanje i razvoj...), sustavi, sredstva za rad (oprema, investicijska dobra) i prava (licence, najmovi, lizing) [14].

3.2. Svrha nabave

Objedinjujući prethodno rečeno, možemo definirati svrhu nabave. Osnovna svrha nabave jest povezati i uskladiti potrebe pripadajuće organizacije za sredstvima, uslugama i energijom koje ne proizvodi sama, sukladno vlastitim interesima te interesima dobavljača, a da se pri tom ne naruši izvođenje ostalih poslovnih procesa [3].

Ispunjujući navedenu svrhu, osnovni cilj poduzeća jest nabavljati potrebne materijale i usluge zadovoljavajući sljedeće kriterije [4]:

- funkcionalno odgovarajuća kakvoća;
- najpovoljnija cijena;
- ekonomična količina;
- najmanji rizici;
- pravovremena i pouzdana dostava;
- očuvanje okoliša;
- očuvanje odnosa s dobavljačima.

Prije kontaktiranja dobavljača poželjno je da svako poduzeće analizom stanja zaliha utvrdi potrebne količine i kakvoću potrebnih materijala. Nakon toga slijedi analiza dobavljača koja rangira iste po preostalim gore navedenim kriterijima, a to su najpovoljnija cijena te mogućnost pravovremene i sigurne dostave uz što veće očuvanje okoliša.

3.3. Načela nabave

U svakom poduzeću proces odnosno funkcija nabave indirektno ili direktno utječe na ostale segmente pripadajućeg poduzeća. Stoga, u svakom poduzeću, poželjno je da se nabava zasniva na četiri temeljna načela [3]:

1. stvaranje vrijednosti i dobiti za organizaciju – vrijednost i dobit ostvaruje se širenjem dosega nabave, održavanjem dugoročnih odnosa s dobavljačima te kontinuiranim ostvarivanjem uštede;
2. procesi i upravljanje – osigurati kontinuirano poboljšanje sveobuhvatnom transformacijom organizacije, njenih procesa i politike, komunikacijom duž cijele organizacije i maksimiziranjem vrijednosti nabave;
3. mjerenje uspješnosti i naplaćivanje – ostvarivanje najveće troškovne učinkovitosti (omjer troškova i dobitaka) pružanjem konkurentne razine usluga, definiranje razine uspješnosti i uzvratna naplata (plaćanje uzvraćanjem usluga), mjerenje učinkovitosti i efikasnosti rada organizacije;
4. potpora sustavu i upravljanje talentom – osigurati pravovremenu opskrbu poslovnog sustava potrebnim sredstvima te oblikovati karijere ljudi davanjem osjećaja više svrhe.

3.4. Problemi nabave

Problemi nabave mogu nastajati zbog unutarnjih i vanjskih faktora. Problemi koji nastaju zbog unutarnjih faktora problemi su na koje poduzeće ima utjecaj i najčešće je samo krivac za nastajanje istih. Problemi nastali zbog vanjskih faktora predstavljaju skupinu problema na koje poduzeće naručitelj nema direktan ili nikakav utjecaj [3].

Glavni razlog nastajanja problema uzrokovanih unutarnjim faktorima jest loše planiranje nabave. Analiza postojeće situacije početna je točka uspješnog odnosno neuspješnog planiranja nabave i ako je ista loše provedena, može rezultirati nekolicinom problema.

Jedan od tih problema jesu nepravovremeni zahtjevi za nabavom koji rezultiraju naručivanjem premalo ili previše dobara, naručivanjem krivih dobara ili kašnjenjem isporuke. Gomilanjem viška robe u skladištu, ovisno o vrsti robe, može doći do propadanja odnosno gubitka vrijednosti iste zbog starenja, manjka potražnje ili neočekivanih promjena cijena na

tržištu. S druge strane, nedostatak materijala u potrebno vrijeme može uzrokovati zastoj u procesu proizvodnje, a posljedica toga jesu povećavani troškovi iste, otežano planiranje procesa prodaje i nemogućnost prodaje dobara u planiranom razdoblju [3].

Osim analize stanja u skladištu, potrebno je provesti i analizu dobavljača, skupiti ponude istih i odabrati najbolje odgovarajuću za vlastito poduzeće. Na odabir ponude utječe više vanjskih faktora, a to su: kvaliteta proizvoda, cijena proizvoda, način plaćanja, transportni troškovi, način pakiranja, mogućnost praćenja pošiljke i kredibilitet dobavljača [3].

Nabavkom materijala odnosno proizvoda loše kvalitete može doći do nakupljanja loše i neupotrebljive robe u skladištu kao i do reklamacija od strane kupaca što povlači dodatne troškove kompenzacije. Nadalje, loš način pakiranja riskira oštećenje robe u transportu ili gubitak iste.

Osim spomenutih potencijalnih problema nabave, postoje i razni vanjski faktori na koje ne mogu imati utjecaj niti naručitelj niti dobavljač. Neki od njih su: prirodne katastrofe, neočekivane gužve u prometu, neočekivani kvarovi transportnih vozila i dr.

3.5. Različiti pristupi u nabavi

Određeni problemi navedeni u prethodnom potpoglavlju više su ili manje karakteristični za različite modele odnosno pristupe u nabavi koji su nastali kroz povijest i na određen način pratili razvitak iste. Ukratko, spomenuti pristupi su [9]:

- EOQ (eng. *Economic Order Quantity*) – ekonomična odnosno optimalna količina narudžbi kojom je cilj minimizirati troškove nabavljanja, zapremanja i skladištenja zaliha. Karakteristično za slučajeve u kojima su spomenuti troškovi konstantni tijekom duljeg vremenskog perioda. Problemi: promjena ponude i potražnje, promjena troškova skladištenja tijekom različitih perioda u godini, različite cijene za narudžbe većih količina;
- MRP (eng. *Material Requirements Planning*) – planiranje potreba za materijalom s ciljem zadovoljenja ovisne potražnje uz minimalno zadržavanje zaliha i uz minimalne troškove. Jedan od prvih pristupa koji integrira informacijski sustav u poslovni te pri tom od ulaznih podataka koristi glavni plan proizvodnje, popis materijala (količina, kvaliteta i vrijeme kada su potrebni) te trenutno stanje zaliha.

Ovisna potražnja predstavlja potražnju više međusobno ovisnih materijala koji kao takvi čine jednu cjelinu odnosno finalni proizvod. MRP upravo prepoznaje tu vezu te rješava problem predviđanja potreba za jednim materijalom, ovisno o drugome. Problemi: nepouzdani programski alati, integritet podataka, kontrola kvalitete, fluktuacije cijena;

- MRP II (eng. *Manufacturing Resource Planning*) – planiranje resursa za proizvodnju nasljednik je planiranja potreba za materijalom, za razliku od kojeg uključuje i podatke vezane uz financijske potrebe kao i potrebe zaposlenika te na taj način povezuje planiranje nabave sa financijskim i marketinškim planovima. Uz samostalno korištenje, često se koristi i kao modul ERP (eng. *Enterprise resource planning*) sustava. Vrijede isti problemi kao kod MRP-a, no MRP II rješava potencijalni problem fluktuacija cijena na način da omogućuje predviđanje raznih ishoda simulacijom određenih scenarija za zadane ulazne varijable;
- JIT (eng. *Just in Time*) – osnova sustava nabave upravo na vrijeme jest da se nabava materijala vrši sukladno potrebama proizvodnje, odnosno netom prije samog procesa proizvodnje. Time se nastoje ukloniti troškovi čuvanja ili propadanja zaliha. Postoje dva oblika JIT sustava – Kanban i Kaizen. Ciljevi Kanban oblika su: uravnoteženo okruženje u kojem se istovremeno ne obavlja više od jednog zadatka, održavanje transparentnosti i vizualizacija tijekom razvoja proizvoda, konstantna prilagodba potrebama i željama kupaca, jačanje kohezije među dijelovima organizacije itd. Temeljni cilj Kaizena jest postići postepeno i kontinuirano poboljšanje kvalitete, fleksibilnosti i produktivnosti uz istovremeno smanjenje troškova. Najčešće korištena metodologija za provedbu navedenog u djelo jest PDCA (eng. *Plan-Do-Check-Act*). Oba oblika dijele nekolicinu sličnosti i međusobno se nadopunjuju, pa jedan bez drugog najčešće i nemaju smisla. Problemi JIT sustava su: kašnjenje dostave od strane dobavljača, nestabilna proizvodnja, iznenadne promjene cijena potrebnih materijala;
- TQM (eng. *Total Quality Management*) – potpuno upravljanje kvalitetom pristup je čiji je temeljni cilj zadovoljiti potrebe korisnika i ojačati odnos sa istima povećanjem kvalitete usluga i proizvoda kontinuiranim unapređenjem internih procesa organizacije. Unapređenje procesa uključuje provođenje raznih metrika za praćenje napretka, donošenje odluka baziranih na činjenicama i održavanje komunikacije među organizacijskim jedinicama s ciljem uključivanja zaposlenika u cijeli proces kako bi se istima pružio osjećaj više svrhe radeći za jedinstveni

zajednički cilj. Problemi: visoki troškovi edukacije, primjene odgovarajućih metoda i konstantne kontrole procesa, povećani stres pri izvođenju procesa, potrebne napredne vještine i predanost svih organizacijskih jedinica za uspješnu primjenu i održavanje pristupa;

- TOC (eng. *Theory of Constraints*) – teorija ograničenja nalaže da je sustav jak onoliko koliko je osjetljiv na vlastitu najslabiju točku. U kontekstu nabave, teorija ograničenja nastoji ukloniti najveća ograničenja koja sprječavaju neometano izvođenje procesa nabave i postizanje zadanih ciljeva. Pri tome, pod ograničenjima se smatraju naslabiji dijelovi sustava, bilo da se radi o ljudima, tehnologiji ili vremenskom aspektu. Problemi: identifikacija ograničenja i uzročno-posljedičnih veza među istima (koja su rješenja kratkotrajna, a koja dugotrajna?) te varijabilni faktori zbog kojih nastaju određena ograničenja (npr. privremena promjena ponude i potražnje);
- TBC (eng. *Time-Based Competition*) – konkurencija temeljena na vremenu pristup je koji zapravo proširuje principe JIT-a. Drugim riječima može se reći da, u vrijeme kada se pojavio, JIT je bio prvi oblik TBC pristupa. Razvitkom prethodno opisanih pristupa, uklonjene su mnoge tehničke prepreke u procesu nabave i proizvodnje te je time presudni faktor koji čini razliku između organizacija postalo vrijeme, odnosno vremenska optimiziranost procesa uz zadržavanje kvalitete proizvoda. TBC pristup tako vrijeme smatra najvažnijim čimbenikom poslovanja, a osnovni cilj jest skratiti protok roba i usluga kroz nabavni lanac. Problemi: ovisnost o pouzdanosti dobavljača, nedostupnost informacija ili materijala u potrebnom trenutku te preostali vanjski faktori navedeni u prethodnom poglavlju na koje poduzeće najčešće ne može imati utjecaja;
- SCM (eng. *Supply Chain Management*) – upravljanje lancem nabave pristup je odnosno proces koji obuhvaća sve podprocese koji transformiraju sirove materijale u finalne proizvode. Umrežavajući organizacije, resurse, aktivnosti i tehnologije uključene u proces nabave nastoje se minimizirati troškovi, povećati prihodi i maksimizirati vrijednost za kupce i dobavljače te time ostvariti prednost u odnosu na konkurentske organizacije. Problemi: neočekivani troškovi transporta uzrokovani promjenama cijena goriva, napredak tehnologije, povećani broj globalnih korisnika, potreba za prebacivanjem proizvodnje u zemlje sa jeftinijom radnom snagom (otežana kontrola lanca nabave).

4. Dinamičko programiranje

Dinamičko programiranje matematički je postupak koji rješava probleme višeetapnih procesa upravljanja pri čemu je osnovni cilj optimizacija istih. Obzirom da je osnovni fokus ovog rada na korištenju metode dinamičkog programiranja u rješavanju problema nabave, u prethodna dva poglavlja su između ostaloga definirani pojmovi zaliha i nabave te prikazani problemi istih. Dinamičko programiranje nudi rješenje većine od navedenih problema, posebice onih vezanih uz planiranje količina robe za određeni period i kao takvo je od izrazite strateške važnosti za svako poduzeće [5].

Osnovna ideja dinamičkog programiranja jest raspodijeliti glavni kompleksni problem na više manjih jednostavnih te pronalaskom njihovih rješenja riješiti početni problem. Za uspješnu primjenu metode dinamičkog programiranja potrebno je da svaki promatrani sustav odnosno proces ima jasno postavljen matematički model te funkciju cilja istoga, bilo da se radi o minimizaciji ili maksimizaciji.

4.1. Dinamičko programiranje i višeetapni procesi

Višeetapni procesi su procesi u kojima se donosi niz odluka, a donose se postepeno i u određenim vremenskim razmacima. Važnost metode dinamičkog programiranja za višeetapne procese leži u činjenici da rješava probleme istih. Primjer toga jest problem nabave što je i osnovni predmet ovog rada. Proces nabave višeetapni je proces koji se ponavlja u određenim intervalima te u svakoj iteraciji uključuje parametre koji mogu, a najčešće i variraju za svako razdoblje. Ti parametri su [2]:

- maksimalna količina zaliha;
- troškovi čuvanja zaliha;
- potražnja;
- maksimalna količina nabave;
- troškovi nabave.

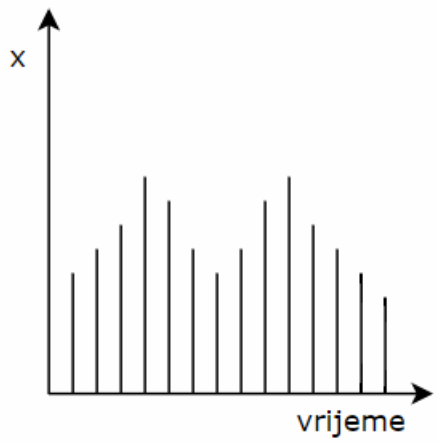
Obzirom na promjenjivu prirodu navedenih parametara, svako sljedeće razdoblje ovisit će o dobivenom rezultatu iz prethodnog razdoblja. Kako bi se izbjeglo ponovno računanje za prethodna razdoblja, koristi se metoda dinamičkog programiranja koja omogućuje pamćenje dobivenih rezultata te njihovo ponovno korištenje u novoj iteraciji [2].

4.1.1. Klasifikacija višeetapnih procesa odlučivanja

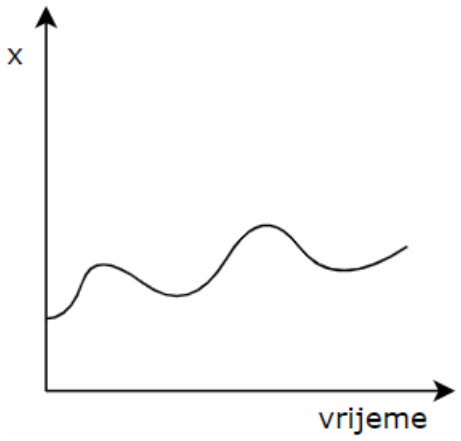
Višeetapni procesi, ovisno o svojim karakteristikama, mogu se podijeliti na dvije podvrste. Pri tome isti se klasificiraju prema dvije karakteristike [2]:

1. Vremenski diskretni i vremenski kontinuirani;
2. Deterministički i probabilistički odnosno vjerojatni.

Kod prve klasifikacije očito je da je vrijeme osnovni faktor o kojem će ovisiti vrijednost varijable. Vremenski diskretne varijable definirane su skupom diskretnih točaka u prostoru, primjerice skup srednjih temperatura mora za određeno područje, recimo Dubrovnik, zapisanih u jednakim vremenskim intervalima. S druge strane, kontinuirani zapis temperature mora bio bi vremenski kontinuirana varijabla. Za bolje razumijevanje navedenoga pogledajmo slike 2. i 3.



Slika 2: Vremenski diskretna varijabla (Prema: Dobrenić, 1978)



Slika 3: Vremenski kontinuirana varijabla (Prema: Dobrenić, 1978)

Kod druge klasifikacije osnovni faktor koji utječe na vrijednost funkcije $y[x]$ jest njezin argument x. Funkcija deterministčkih višeetapnih procesa potpuno ovisi o pripadajućem argumentu. Primjer takve funkcije je funkcija prihoda, pod uvjetom da je odnos između uloženog i dobivenog novca točno poznat.

Ipak, u stvarnosti, zbog raznih nepredvidivih faktora (promjena potražnje uzrokovana novim trendovima, vremenskim neprilikama i sl.), teško je predvidjeti iznos prihoda, stoga se koriste nasumične (eng. *random*) varijable. Takva vrsta funkcije čije vrijednosti ovise o nasumičnim varijablama naziva se probabilistička funkcija.

Procesi koji se mogu prikazati isključivo determinističkim funkcijama, nazivaju se deterministički procesi. Ukoliko prikaz procesa sadrži jednu ili više probabilističkih funkcija, tada se radi o probabilističkom ili vjerojatnom procesu [2].

Obzirom da formuliranje modela te rješavanje problema za vremenski kontinuirane i probabilističke višeetapne procese ne spada u domenu dinamičkog programiranja, u nastavku će se promatrati samo vremenski diskretni i deterministički višeetapni procesi odlučivanja.

4.2. Algoritam dinamičkog programiranja

Pored opisanog u prethodnom poglavlju, dinamičko programiranje svoju primjenu pronalazi u raznim područjima i često je osnova drugih algoritama poput Bellman-Fordovog, Floyd-Warshallovog, Kadanovog i drugih [11].

Algoritam dinamičkog programiranja za rješenje problema nabave uključuje jednostavne matematičke operacije čiji su ulazni parametri već nabrojani u poglavlju prije, a to su: maksimalna količina zaliha, troškovi čuvanja zaliha, potražnja za zadano razdoblje, maksimalna količina nabave i troškovi nabave.

Kod zadavanja problema nabave potrebno je utvrditi da isti zadovoljava sljedeće postavke [2]:

- nabava u vremenu koje prethodi promatranom razdoblju je nula;
- početak procesa nabave uvjetovan je početkom razdoblja;
- na početku prvog i na kraju posljednjeg razdoblja količina zaliha na skladištu iznosi nula;
- potražnja se javlja na početku razdoblja;

- nedostatak robe na zalihi nije dopušten.

Prije prikaza samog algoritma na zadanom primjeru, potrebno je biti upoznat sa određenim označavanjima objašnjenima u nastavku.

4.2.1. Označavanje u algoritmu dinamičkog programiranja

Iako se, kako je prethodno rečeno, algoritam dinamičkog programiranja sastoji od jednostavnih matematičkih operacija, označavanja upotrebljena za formulaciju potrebnih modela mogu se činiti poprilično kompliciranima. Kako bi se mogli u potpunosti razumjeti nadolazeći primjeri primjene algoritma dinamičkog programiranja, u nastavku su definirana ključna označavanja korištena u istima [2]:

- Funkcija f – predstavlja funkciju uspjeha smanjenja troškova koju je cilj minimizirati
- okrugle zagrade ( ) – označavaju vrijednost vremena za vremenski zavisne varijable. Primjerice $x(i)$ bi predstavljalo „vrijednost vremenski zavisne varijable x u vremenu i;
- uglate zagrade [ ] – označavaju argument neke druge funkcionalne povezanosti. Primjerice $y[x_1,\ x_2]$ bi značilo da je količina y funkcija varijabli x_1 i x_2. Tako bi jednadžba

  $$H[i] = H[x(i) - u(i)]$$

  u prijevodu značila „vrijednost od H u vremenu i je funkcija razlike $(x - y)$ u vremenu i“;
- vitičaste zagrade { } – vežu se uz maksimum ili minimum. Primjerice, jednadžba

  $$\min \{ y\,[x_1, x_2] \}$$

  bi značila „minimalna vrijednost od $y[x_1, x_2]$ koja se može dobiti podešavanjem x_1 i x_2 bez kršenja nekog određenog ograničenja;

- brojevi $1, 2, \dots, N$ – predstavljaju broj etape odnosno vremenskog perioda za koji je izvršen ili se u promatranom trenutku vrši izračun;
- brojevi $N, N-1, N-2, \dots, 2, 1$ – broj vremenskog perioda od kojeg se polazi odnosno perioda čije se vrijednosti koriste za izračun vrijednosti u tekućem periodu.

Primjer:

Godina / i	Broj etape od koje se polazi
1	N
2	$N-1$
3	$N-2$
.	.
.	.
.	.
$N-1$	2
N	1

Pored navedenih značenja za svaku od zagrada, bitno je napomenuti da zagrade također mogu imati i običnu funkciju u tekstu zadatka ili u algebarskim izrazima. U svakom slučaju, ovisno o kontekstu u kojem se koristi, bit će moguće zaključiti o kojoj ulozi zagrade je riječ [2].

4.2.2. Načelo optimalnosti

Računski postupak korišten u algoritmu dinamičkog programiranja temelji se na načelu optimalnosti koje je zapravo ključ uspješnosti dinamičkog programiranja. Načelo optimalnosti primjenjivo je na sve višeetapne procese odlučivanja, a glasi [2]:

„Optimalna politika ima svojstvo, da ma kakvo bilo početno stanje c i optimalna prva odluka u prvom vremenskom periodu, daljnje odluke moraju činiti optimalnu politiku s obzirom na stanje koje rezultira iz prve odluke.“

Poštivajući načelo optimalnosti, postupak dinamičkog programiranja uključivat će sljedeće korake [2]:

- definiranje funkcije optimalnog prihoda odnosno troškova;
- izvođenje funkcionalne jednadžbe za funkciju optimalnog prihoda odnosno troškova;
- korištenje funkcionalne jednadžbe kako bi se pronašle funkcije optimalne odluke koje daju optimalnu politiku.

U nadolazećem primjeru zadatka prikazan je algoritam dinamičkog programiranja pri čemu su provedeni navedeni koraci.

4.2.3. Teorijski izvod algoritma dinamičkog programiranja

Prije same primjene algoritma na priloženom primjeru, potrebno je prikazati proces formulacije modela problema nabave. Prethodno spomenutim ulaznim parametrima, pri formuliranju modela problema nabave dodjeljuju se sljedeće oznake [2]:

- $I(i)$ – količina zaliha na kraju razdoblja i;
- $Q(i)$ – količina nabave na početku razdoblja i;
- $D(i)$ – količina potražnje na početku razdoblja i;
- $C_p(i)$ – troškovi nabave za razdoblje i;
- $C_h(i)$ – troškovi čuvanja zaliha za razdoblje i.

Temeljni cilj rješavanja problema nabave jest minimizirati ukupne troškove za cijelo planirano vremensko razdoblje kroz N perioda odnosno etapa. Navedena minimizacija prikazuje se sljedećim matematičkim izrazom [2]:

$$C_t = \sum_{n=i}^{N} \left[C_p(i) + C_h(i) \right] \rightarrow min.$$

Ako se na početku promatranog razdoblja i nabavi količina $Q(i)$, na kraju razdoblja količina zaliha biti će $I(i)$. Matematički izraz za rečeno je [2]:

$$I(i) = I(i-1) + Q(i) - D(i).$$

Prilagodbom tog izraza iz navedenog se dobiva da se zalihe iz prethodnog razdoblja mogu izračunati koristeći formulu

$$I(i-1) = I(i) + D(i) - Q(i).$$

Što se zaliha tiče, njihova maksimalna količina u svakom problemu eksplicitno je zadana, a obzirom da logički gledano zalihe ne mogu biti manje od nule, rubne količine zaliha definiraju se sljedećim izrazom:

$$0 \leq I(i-1) \leq maks.kol.zaliha.$$

Uvrštavanjem desne strane prethodne jednadžbe, umjesto $I(i-1)$ uvrštavamo $I(i) + D(i) - Q(i)$ te dobivamo

$$0 \leq I(i) + D(i) - Q(i) \geq maks.kol.zaliha.$$

Kako bi se iz zadnje relacije dobila relacija za određivanje rubnih količina nabave $Q(i)$, potrebno je od svakog dijela gornje relacije oduzeti izraz $[I(i) + D(i)]$ te potom sve pomnožiti sa (-1). Time dolazimo do ključne formule algoritma dinamičkog programiranja za rješavanje problema nabave koja glasi:

$$I(i) + D(i) - maks.kol.zaliha \leq Q(i) \geq I(i) + D(i).$$

Ukupni troškovi procesa nabave u promatranom razdoblju ovisit će o količinama nabave u tom razdoblju, prijenosnim zalihama za sljedeće razdoblje te o troškovima uzrokovanim odlukama u prethodnim razdobljima. Iz toga proizlazi da se ukupni troškovi procesa nabave koji traje N razdoblja mogu izraziti sljedećom funkcijom [2]:

$$g(N)[Q(N), I(N)] + f(N-1)[I(N-1)].$$

Pri odabiru optimalnog izbora za $Q(N)$ bira se onaj koji minimizira gornju funkciju. Na osnovu toga načelo optimalnosti daje sljedeću funkcionalnu jednadžbu kojom se pronalazi vektor količina koji minimizira ukupne troškove za predviđenu količinu prodaje [2]:

$$f(N)\ [I(N)] = min\{\ g(N)\ [Q(N), I(N)] + f(N-1)\ [I(N-1)]\}.$$

Ako umjesto $I(N-1)$ sa kraja dobivene funkcionalne jednadžbe uvrstimo desnu stranu izraza $I(i-1) = I(i) + D(i) - Q(i)$, uz zamjenu i sa N, tada će finalna verzija navedene funkcionalne jednadžbe glasiti:

$$f(N)\ [I(N)] = min\{\ g(N)\ [Q(N), I(N)] + f(N-1)\ [I(N) + D(N) - Q(N)\ ]\}.$$

4.2.4. Primjena algoritma na zadanom primjeru

Poduzeće X treba nabaviti sirove materijale potrebne za proizvodnju gotovih proizvoda. U svakom ciklusu nabave nabavlja se po 10 komada, sa maksimalnom količinom nabave od 50 komada. Troškovi jedne nabave iznose 500 kuna te je bitno uzeti u obzir da je maksimalna količina materijala koje je moguće zapremiti 20 komada. Potrebno je izraditi optimalni plan nabave kroz 3 razdoblja i izračunati ukupne troškove upravljanja zalihama (trošak nabave + trošak čuvanja zaliha), ako trošak skladištenja za promatrani materijal iznosi 4 kune po jedinici, a predviđena potražnja po periodima jest 20, 30, 20. Cilj izrade plana je moći odrediti optimalne količine zaliha sirovih materijala i predvidjeti ukupni trošak nabavnog procesa.

Osnovna formula pomoću koje se računaju količine robe za nabavu:

$$zalihe(i) + potražnja(i) - maks.kol.zaliha \leq nabava(i) \leq zalihe(i) + potražnja(i)$$

Pri izračunu vrijednosti koristeći navedenu formulu, bitno se sjetiti pravila da nabava nikada ne može biti manja od 0 ili veća od zadane maksimalne količine, koja u ovom primjeru iznosi 50.

Primjenom formule, za nabavu u prvom razdoblju određuju se granične vrijednosti iste:

$$20 + 20 - 20 \leq nabava\ (1) \leq 20 + 20$$

$$20 \leq nabava\ (1) \leq 40$$

Bitno je napomenuti da se kod primjene formule u prvom razdoblju u zalihe uvrštava maksimalna količina zalihe, dok se u nadolazećim razdobljima u zalihe uvrštava količina za koju se provodi račun u trenutnom koraku računanja za to razdoblje. Obzirom da maksimalna količina zaliha koje se mogu držati na skladištu iznosi 20 komada, a nabava se vrši u ratama od 10 komada, postoje tri moguća stanja na skladištu, točnije 0, 10 i 20 komada. Sukladno tome, za prvo razdoblje provodi se sljedeći izračun:

$$f(i)\ [I(1)] = \min\ \{g(1)\ [Q(1), I(1)]\} = cijena\ nabave + I(1) \cdot tr.zaliha\ po\ jedinici$$

$$f(1)[0]\ = g(1)[20,0]\ = 500 + 0 \cdot 4\ = 500$$
$$f(1)[10] = g(1)[30,10] = 500 + 10 \cdot 4 = 540$$
$$f(1)[20] = g(1)[40,20] = 500 + 20 \cdot 4 = 580$$

Dobivene vrijednosti upisuju se u tablicu rješenja problema nabave:

Tablica 1: Prikaz rješenja problema nabave za prvo razdoblje

I	$Q(1)$	$f(1)$	$Q(2)$	$f(2)$	$Q(3)$	$f(3)$
0	20	500				
10	30	540				
20	40	580				

(Izvor: izrada autora rada)

U svakom razdoblju nakon prvog, računaju se minimalni troškovi narudžbi za svaku količinu zaliha (0, 10 i 20). Za razliku od prvog razdoblja, izračun uključuje i treći broj (580, 540, 500) koji se pronalazi u prethodnom razdoblju na križanju $f(i)$ i količine zaliha I dobivene u drugim uglatim zagradama. Matematički izraz navedenoga glasi:

$$f(2)\,[I(2)] = \min\,\{g(2)\,[Q(2), I(2)] + f(1)\,[I(2) + D(2) - Q(2)]\}$$

Tako u drugom razdoblju u kojem potražnja iznosi 30 komada, primjenom dobivene formule u koraku rješavanja za svaku od varijanti količina zaliha dobiva se:

1. Korak:

$$0 + 30 - 20 \leq nabava\,(2) \leq 0 + 30$$

$$10 \leq nabava\,(2) \leq 30$$

$$f(2)[0] \rightarrow \min \begin{cases} g(2)[10,0] + f(1)[0 + 30 - 10] = 500 + 0 \cdot 4 + 580 = 1080 \\ g(2)[20,0] + f(1)[0 + 30 - 20] = 500 + 0 \cdot 4 + 540 = 1040 \\ g(2)[30,0] + f(1)[0 + 30 - 30] = 500 + 0 \cdot 4 + 500 = \mathbf{1000} \end{cases}$$

---

2. Korak:

$$10 + 30 - 20 \leq nabava\,(2) \leq 10 + 30$$

$$20 \leq nabava\,(2) \leq 40$$

$$f(2)[10] \rightarrow \min \begin{cases} g(2)[20,10] + f(1)[10 + 30 - 20] = 500 + 10 \cdot 4 + 580 = 1120 \\ g(2)[30,10] + f(1)[10 + 30 - 30] = 500 + 10 \cdot 4 + 540 = 1080 \\ g(2)[40,10] + f(1)[10 + 30 - 40] = 500 + 10 \cdot 4 + 500 = \mathbf{1040} \end{cases}$$

---

3. Korak:

$$20 + 30 - 20 \leq nabava\,(2) \leq 20 + 30$$

$$30 \leq nabava\,(2) \leq 50$$

$$f(2)[20] \rightarrow \min \begin{cases} g(2)[30,20] + f(1)[20 + 30 - 30] = 500 + 20 \cdot 4 + 580 = 1160 \\ g(2)[40,20] + f(1)[20 + 30 - 40] = 500 + 20 \cdot 4 + 540 = 1120 \\ g(2)[50,20] + f(1)[20 + 30 - 50] = 500 + 20 \cdot 4 + 500 = \mathbf{1080} \end{cases}$$

Na kraju razdoblja ponovno se popunjava tablica rješenja problema nabave dobivenim vrijednostima u svakom koraku. Dobiveni minimalni troškovi nabave upisuju se u $f(i)$ stupac, a količina nabave u retku dobivenih minimalnih troškova upisuje se u $Q(i)$ stupac.

Tablica 2: Prikaz rješenja problema nabave za drugo razdoblje

I	$Q(1)$	$f(1)$	$Q(2)$	$f(2)$	$Q(3)$	$f(3)$
0	20	500	30	1000		
10	30	540	40	1040		
20	40	580	50	1080		

(Izvor: izrada autora rada)

Isti postupak provodi se za treće razdoblje, osim što se gledaju rezultati dobivenu u drugom, a ne prvom razdoblju. U trećem razdoblju potražnja iznosi 20, stoga računamo:

1. Korak:

$$0 + 20 - 20 \leq nabava\ (3) \leq 0 + 20$$

$$0 \leq nabava\ (3) \leq 20$$

$$f(3)[0] \rightarrow \min \begin{cases} g(3)[0,0] + f(2)[0 + 20 - 0] = 0 + 0 \cdot 4 + 1080 = \mathbf{1080} \\ g(3)[10,0] + f(2)[0 + 20 - 10] = 500 + 0 \cdot 4 + 1040 = 1540 \\ g(3)[20,0] + f(2)[0 + 20 - 20] = 500 + 0 \cdot 4 + 1000 = 1500 \end{cases}$$

---

2. Korak:

$$10 + 20 - 20 \leq nabava\ (3) \leq 10 + 20$$

$$10 \leq nabava\ (3) \leq 30$$

$$f(3)[10] \rightarrow \min \begin{cases} g(3)[10,10] + f(2)[10 + 20 - 10] = 500 + 10 \cdot 4 + 1080 = 1620 \\ g(3)[20,10] + f(2)[10 + 20 - 20] = 500 + 10 \cdot 4 + 1040 = 1580 \\ g(3)[30,10] + f(2)[10 + 20 - 30] = 500 + 10 \cdot 4 + 1000 = \mathbf{1540} \end{cases}$$

---

3. Korak:

$$20 + 20 - 20 \leq nabava\ (3) \leq 20 + 20$$

$$20 \leq nabava\ (3) \leq 40$$

$$f(3)[20] \rightarrow \min \begin{cases} g(3)[20,20] + f(2)[20 + 20 - 20] = 500 + 20 \cdot 4 + 1080 = 1660 \\ g(3)[30,20] + f(2)[20 + 20 - 30] = 500 + 20 \cdot 4 + 1040 = 1620 \\ g(3)[40,20] + f(2)[20 + 20 - 40] = 500 + 20 \cdot 4 + 1000 = \mathbf{1580} \end{cases}$$

Finalna verzija tablice rješenja problema nabave sadrži sljedeće vrijednosti:

Tablica 3: Finalna verzija tablice rješenja problema nabave

I	$Q(1)$	$f(1)$	$Q(2)$	$f(2)$	$Q(3)$	$f(3)$
0	20	500	30	1000	0	1080
10	30	540	40	1040	30	1540
20	40	580	50	1080	40	1580

(Izvor: izrada autora rada)

Kako bismo dobili ukupne troškove nabave te troškove čuvanja zaliha, odnosno ukupne troškove cijelog nabavnog procesa, potrebno je popuniti sljedeću tablicu koristeći se formulom:

$$I(i-1) = I(i) + D(i) - Q(i)$$

pri čemu je od prije poznato da nepoznanice predstavljaju sljedeće:

- I – zalihe;
- D – potražnja;
- Q – nabava.

Tablica 4: Početna verzija tablice kretanja količina i ukupnih troškova

Razdoblje (i)	Zalihe $I(i-1)$	Nabava $Q(i)$	Potražnja $D(i)$	Zalihe $I(i)$	Troškovi	
					tr. nabave $Cp(i)$	tr. zaliha $Ch(i)$
1	0		20			
2			30			
3			20	0		
			Ukupno			
			Sveukupno			

(Izvor: izrada autora rada)

Početne vrijednosti tablice prikazane su iznad. Popisuje se redni broj razdoblja, potražnja po razdobljima te zalihe za razdoblje prije prvog i za treće razdoblje koje će uvijek

iznositi 0. Stupac $Q(i)$ popunjava se tako da se prepisuju vrijednosti dobivene u tablici rješenja problema nabave. Preciznije, uzimaju se u obzir broj razdoblja i količine zaliha u tom razdoblju. Tako u tablici na križanju nabave za treće razdoblje i zaliha količine 0, nabava također iznosi 0. Sada kada je poznata količina zaliha $I(i)$, potražnja $D(i)$ i nabava $Q(i)$, primjenjuje se prethodno spomenuta formula $I(i-1) = I(i) + D(i) - Q(i)$ te se dobiva rezultat 20. Obzirom da taj broj predstavlja količine zaliha za prethodno razdoblje $I(i-1)$, taj broj prepisuje se u zalihe $I(i)$ prethodnog razdoblja, u ovom slučaju drugog. Ovaj postupak ponavlja se dok god je to potrebno, odnosno sve dok se ne izračunaju zalihe $I(i)$, nabava Q(i), potražnja $D(i)$ te zalihe prethodnog razdoblja $I(i-1)$ za svako od promatranih razdoblja. Drugi korak označen je svijetlo crvenom, a treći svijetlo zelenom bojom.

Tablica 5: Finalna verzija tablice kretanja količina i ukupnih troškova

Razdoblje (i)	Zalihe $I(i-1)$	Nabava $Q(i)$	Potražnja $D(i)$	Zalihe $I(i)$	Troškovi	
					tr. nabave $Cp(i)$	tr. zaliha $Ch(i)$
1	0	20	20	0	500	-
2	0	50	30	20	500	80
3	20	0	20	0	-	-
			Ukupno		1000	80
			Sveukupno		1080	

(Izvor: izrada autora rada)

Nakon izračuna kretanja količina tijekom promatranih razdoblja, izračunavaju se trošak nabave $Cp(i)$ i trošak zaliha $Ch(i)$ pojedinih razdoblja. Za trošak nabave u obzir se uzima cijena jedne nabave koja u ovom primjeru iznosi 500. Trošak zaliha računa se tako da se količina zaliha za promatrano razdoblje pomnoži sa troškom čuvanja zaliha po jedinici, konkretno u ovom primjeru 4kn/jedinici. Ako količine nabave ili zaliha iznose 0, logično je da je i trošak istih 0. Na samom kraju troškovi nabave i zaliha po razdobljima kumulativno se zbrajaju čime se dolazi do ukupnih troškova nabavnog procesa koji iznose 1080 kuna.

5. Profil tvrtke

Tvrtka čiji će se podaci u nastavku koristiti za prikaz primjene dinamičkog programiranja na stvarnom primjeru jest Q d.o.o. (u nastavku samo Q). Globalna agencija Q privatnog je vlasništva te je jedna od najbrže rastućih tehnoloških kompanija u Europi. Osnovana je 2013. godine od strane trojice entuzijasta koji su nakon dugogodišnjeg rada za inozemne kompanije odlučili pokrenuti i osnovati svoju razvojnu tvrtku (eng. *start-up*) u Hrvatskoj. Cijela priča u početku je začeta kao hobi odnosno sporedni posao osnivača, a tek krajem 2015. godine, današnji glavni izvršni direktor (eng. *Chief Executive Officer CEO*), glavni tehnički direktor (eng. *Chief Technology Officer CTO*) te glavni direktor proizvodnje (eng. *Chief Operating Officer COO*) odlučuju se u potpunosti posvetiti vlastitoj tvrtki čemu je slijedila reorganizacija i optimizacija procesa, a svo poslovanje usmjereno je ka stranim klijentima. [Izvor: Šandrk P., Cindrić F., osobni intervju, 10. srpnja 2019]

Danas tvrtka zapošljava i surađuje s ukupno 120 stalno zaposlenih unutarnjih i vanjskih suradnika diljem 20 zemalja svijeta, a uslugu pruža klijentima od New Yorka do Dubaia. Sjedište tvrtke nalazi se u Zagrebu, a još 5 ureda nalazi se u New Yorku, Los Angelesu, Belfastu, Zürichu i Oslu. Samo neki od poznatih klijenata s kojima tvrtka surađuje su: Coca-Cola, BMW, BBC, Lufthansa, IBM, Visa, Audi, Volkswagen, Vaillant, Facebook, Nestle itd [10].

5.1. Djelatnost i usluge tvrtke

Djelovanje tvrtke projektno je orijentirano, stoga nudi široku domenu usluga koje podržavaju razvoj proizvoda od ideje do realizacije. Te usluge su: digitalni marketing, korisnička podrška, brendiranje (eng. *Branding*), dizajn, razvoj web i mobilnih aplikacija, izrada kompleksnih rješenja te općenito integracija informacijskih i poslovnih sustava [10].

Svaki proces razvoja prototipova odnosno proizvoda od ideje do realizacije uključuje sljedeće korake opisane u nastavku [10]:

- Inicijalni sastanak i radionica (eng. *Product workshop*) – sastanak sa klijentom, iznošenje osnovnih ideja, želja i potreba vezanih uz razvitak i finalnu verziju proizvoda;

- Specifikacija zahtjeva (eng. *Specification Development*) – definiranje procesa i osnovnih zahtjeva te procjena potrebnog vremena i potrebnih resursa za ispunjenje zahtjeva i realizaciju proizvoda;
- Customer Journey & UX (eng. *Customer Journey & UX*) – proučavanje i analiza korisničkih zahtjeva i očekivanja, izrada dijagrama slučaja korištenja i osnovnog nacrta finalnog proizvoda;
- Brendiranje i strategija (eng. *Branding & Strategy*) – definiranje postupaka i razvoj strategije za uspješno plasiranje proizvoda odnosno usluge na tržište, čineći je prepoznatljivom za krajnjeg korisnika;
- Dizajn korisničkog sučelja (eng. *Design & UI*) – dizajniranje korisničkog sučelja, definiranje krajnjeg stvarnog izgleda proizvoda;
- Razvoj softvera (eng. *Software Development*) – analiza problema, razvoj konkretnog proizvoda, održavanje i otklanjanje pogrešaka, optimizacija sustava;
- Vođenje projekta (eng. *Project Management*) – održavanje komunikacije s klijentima, definiranje i delegacija zadataka, upravljanje resursima, upravljanje kvalitetom;
- Održavanje i podrška (eng. *Maintenance & Support*) – održavanje proizvoda, otklanjanje pogrešaka, korisnička podrška;
- Online prodaja (eng. *Online Sales*) – oglašavanje, pronalazak kupaca, prezentacija proizvoda, isticanje prednosti u odnosu na konkurente, definiranje osnovnih uvjeta prodaje i dogovaranje cijene;
- Vođenje društvenih mreža (eng. *Community Management*) – kreiranje sadržaja, komunikacija sa korisnicima, oglašavanje, ažuriranje događanja;
- Isporuka finalnog proizvoda (eng. *Final Product Delivery*) – isporuka tehničke i korisničke dokumentacije, edukacija zaposlenika, isporuka proizvoda odnosno usluge.

6. Opis stvarnog problema

Stvarni problem koji će se rješavati u ovom poglavlju jest problem nabave vode koja se koristi u aparatima za vodu. Obzirom da su zaposlenici tvrtke najveći potrošači iste, broj zaposlenika osnovno je mjerilo po kojem se donose odluke o potrebnim količinama vode za određena vremenska razdoblja. Od ukupnih 120 zaposlenika, oko 80 zaposlenika svoj posao obavlja u uredima tvrtke u Zagrebu. Bilo kako bilo, postoje kraća ili dulja vremenska razdoblja kada broj zaposlenika u uredima varira.

Kako tvrtka nudi opciju rada od kuće odnosno rada van ureda (eng. *Remotely*), manji dio zaposlenika stalno koristi tu privilegiju, dok ostatak zaposlenika samo povremeno. Rad od kuće te odlazak zaposlenika na godišnji odmor osnovni su faktori koji utječu na broj zaposlenika u uredima. Učestalost korištenja godišnjeg odmora te spomenute privilegije varira ovisno o raznim faktorima, kao što su blagdani, produženi vikendi, ljetni mjeseci i sl. Također, pored broja zaposlenika, bitno je imati na umu da će potrošnja vode biti veća tijekom vrućih godišnjih razdoblja.

Trenutni proces nabave vode u poduzeću ne uključuje planiranje niti ikakve složenije procedure, već se voda nabavlja u trenutku kada se primijeti potencijalni nedostatak iste. Budući da proces dostave traje određeno vrijeme te je trajanje istog ponekad nepredvidivo zbog vanjskih faktora na koje dobavljač ne može imati utjecaja, nerijetko dolazi do situacije da određeni vremenski period vode nedostaje na zalihama. Primjena metode dinamičkog programiranja idealan je alat za sprječavanje nastajanja navedenog problema.

6.1. Dobivanje podataka

Korišteni podaci u ovom problemu dobiveni su na osnovu podataka iz kartice partnera za prethodne periode naručivanja vode. Obzirom na osjetljivost podataka te činjenicu da kartica partnera sadrži dio podataka nerelevantnih za rješavanje problema nabave (šifra naloga, ime dobavljača, datum dospijeća, opis knjiženja i dr.), pojednostavljeni oblik kartice partnera sadržavat će sljedeće podatke:

- datum radnog naloga (narudžbe);
- cijena po komadu;
- količina.

Voda se naručuje po komadima (pakovanje u bocama od 18,9 litara) te ima fiksnu cijenu 33,75 kuna. U tablici ispod nalazi se detaljniji prikaz nabavljanja vode za 5 razdoblja.

Tablica 6: Pojednostavljena verzija kartice partnera

Datum radnog naloga	Cijena po komadu (kn)	Količina
2. svibnja 2019.	33,75	30
16. svibnja 2019.	33,75	15
31. svibnja 2019.	33,75	30
15. lipnja 2019.	33,75	15
2. srpnja 2019.	33,75	45

(Izvor: izrada autora rada)

Za problem nabave, količine nabave za navedene datume prikazane u danoj tablici mogu se promatrati kao količine potražnje na početku tog razdoblja. Sudeći po odnosima priloženih količina da se zaključiti da nabavna rata iznosi 15 komada. Iako su priložene cijene po komadu pakovanja, zbog prirode algoritma dinamičkog programiranja cijena nabave neće ovisiti o navedenim cijenama. Tako ukupni troškovi odnosno cijena jedne nabave uključuje samo fiksne troškove kod svake nabave, koji iznose 250,00 kn. Obzirom na prirodu djelatnosti kojom se tvrtka bavi, veći dio prostorija iskorišten je za urede, stoga je skladišni prostor ograničen na 30 komada, a po dobivenoj informaciji od računovođe, trošak čuvanja zaliha iznosi 1,5 kn. Zbog određenih ograničenja dostavnih vozila dobavljača, maksimalna količina nabave iznosi 60 komada. Vodeći se činjenicom da se voda do sada nabavljala u prosjeku svako dva tjedna, sljedeća 4 razdoblja za koja će se računati ukupni troškovi nabave su sredina srpnja, početak i sredina kolovoza te početak rujna. Predviđena potražnja za navedena 4 razdoblja iznosi 30, 15, 45 i 30 komada, respektivno. Smanjena potražnja na početku drugog razdoblja uzrokovana je odlaskom zaposlenika na kolektivni godišnji odmor koji počinje 1. kolovoza svake godine, dok je povećana potražnja u trećem razdoblju uzrokovana vraćanjem zaposlenika s kolektivnog godišnjeg odmora.

6.1. Rješavanje problema

U prethodnom poglavlju prikazano je dobivanje podataka za stvarni problem. Podaci relevantni za primjenu algoritma dinamičkog programiranja su:

- Maksimalna količina zaliha – 30 komada;
- Maksimalna količina nabave – 60 komada;
- Cijena nabave – 250,00 kn;
- Nabavna rata – 15 komada;
- Trošak čuvanja zaliha po jedinici – 1,5 kn;
- Predviđena potražnja za 4 perioda – 30, 15, 45 i 30 komada.

Budući da je postupak primjene algoritma dinamičkog programiranja objašnjen na prethodnom primjeru, u nastavku se nalaze samo rezultati za svaki od provedenih koraka.

1. Razdoblje:

$$30 + 30 - 30 \leq nabava\ (1) \leq 30 + 30$$

$$30 \leq nabava\ (1) \leq 60$$

$$f(i)\ [I(1)] = \min\ \{g(1)\ [Q(1), I(1)]\} = cijena\ nabave + I(1) \cdot tr.zaliha\ po\ jedinici$$

$$f(1)[0]\ = [30,0]\ = 250 + 0 \cdot 1{,}5\ = 250$$
$$f(1)[15] = [45,15] = 250 + 15 \cdot 1{,}5 = 272{,}5$$
$$f(1)[30] = [60,30] = 250 + 30 \cdot 1{,}5 = 295$$

Tablica 7: Prikaz rješenja stvarnog problema nabave za prvo razdoblje

I	$Q(1)$	$f(1)$	$Q(2)$	$f(2)$	$Q(3)$	$f(3)$	$Q(4)$	$f(4)$
0	30	250						
15	45	272,5						
30	60	295						

(Izvor: izrada autora rada)

2. Razdoblje:

$$f(2)\,[I(2)] = \min\,\{g(2)\,[Q(2), I(2)] + f(1)\,[I(2) + D(2) - Q(2)]\}$$

1. Korak:

$$0 + 15 - 30 \leq nabava\,(2) \leq 0 + 15$$

$$0 \leq nabava\,(2) \leq 15$$

$$f(2)[0] \rightarrow \min \begin{cases} [0,0] + [0 + 15 - 0] = 0 + 0 \cdot 1{,}5 + 272{,}5 = \mathbf{272,5} \\ [15,0] + [0 + 15 - 15] = 250 + 0 \cdot 1{,}5 + 250 = 500 \end{cases}$$

---

2. Korak:

$$15 + 15 - 30 \leq nabava\,(2) \leq 15 + 15$$

$$0 \leq nabava\,(2) \leq 30$$

$$f(2)[15] \rightarrow \min \begin{cases} [0,15] + [15 + 15 - 0] = 0 + 15 \cdot 1{,}5 + 295 = \mathbf{317,5} \\ [15,15] + [15 + 15 - 15] = 250 + 15 \cdot 1{,}5 + 272{,}5 = 545 \\ [30,15] + [15 + 15 - 30] = 250 + 15 \cdot 1{,}5 + 250 = 522{,}5 \end{cases}$$

---

3. Korak:

$$30 + 15 - 30 \leq nabava\,(2) \leq 30 + 15$$

$$15 \leq nabava\,(2) \leq 45$$

$$f(2)[30] \rightarrow \min \begin{cases} [15,30] + [30 + 15 - 15] = 250 + 30 \cdot 1{,}5 + 295 = 590 \\ [30,30] + [30 + 15 - 30] = 250 + 30 \cdot 1{,}5 + 272{,}5 = 567{,}5 \\ [45,30] + [30 + 15 - 45] = 250 + 30 \cdot 1{,}5 + 250 = \mathbf{545} \end{cases}$$

Tablica 8: Prikaz rješenja stvarnog problema nabave za drugo razdoblje

I	$Q(1)$	$f(1)$	$Q(2)$	$f(2)$	$Q(3)$	$f(3)$	$Q(4)$	$f(4)$
0	30	250	0	272,5				
15	45	272,5	0	317,5				
30	60	295	45	545				

(Izvor: izrada autora rada)

3. Razdoblje:

$$f(3)\,[I(3)] = \min\,\{g(3)\,[Q(3), I(3)] + f(2)\,[I(3) + D(3) - Q(3)]\}$$

1. Korak:

$$0 + 45 - 30 \leq nabava\,(3) \leq 0 + 45$$

$$15 \leq nabava\,(3) \leq 45$$

$$f(3)[0] \rightarrow \min \begin{cases} [15,0] + [0 + 45 - 15] = 250 + 0 \cdot 1{,}5 + 545 = 795 \\ [30,0] + [0 + 45 - 30] = 250 + 0 \cdot 1{,}5 + 317{,}5 = 567{,}5 \\ [45,0] + [0 + 45 - 45] = 250 + 0 \cdot 1{,}5 + 272{,}5 = \mathbf{522,5} \end{cases}$$

---

2. Korak:

$$15 + 45 - 30 \leq nabava\,(3) \leq 15 + 45$$

$$30 \leq nabava\,(3) \leq 60$$

$$f(3)[15] \rightarrow \min \begin{cases} [30,15] + [15 + 45 - 30] = 250 + 15 \cdot 1{,}5 + 545 = 817{,}5 \\ [45,15] + [15 + 45 - 45] = 250 + 15 \cdot 1{,}5 + 317{,}5 = 590 \\ [60,15] + [15 + 45 - 60] = 250 + 15 \cdot 1{,}5 + 272{,}5 = \mathbf{545} \end{cases}$$

---

3. Korak:

$$30 + 45 - 30 \leq nabava\,(3) \leq 30 + 45$$

$$45 \leq nabava\,(3) \leq 60$$

$$f(3)[30] \rightarrow \min \begin{cases} [45{,}30] + [30 + 45 - 45] = 250 + 30 \cdot 1{,}5 + 545 = 840 \\ [60,30] + [30 + 45 - 60] = 250 + 30 \cdot 1{,}5 + 317{,}5 = \mathbf{612,5} \end{cases}$$

Tablica 9: Prikaz rješenja stvarnog problema nabave za treće razdoblje

I	$Q(1)$	$f(1)$	$Q(2)$	$f(2)$	$Q(3)$	$f(3)$	$Q(4)$	$f(4)$
0	30	250	0	272,5	45	522,5		
15	45	272,5	0	317,5	60	545		
30	60	295	45	545	60	612,5		

(Izvor: izrada autora rada)

4. Razdoblje:

$$f(4)\ [I(4)] = \min\ \{g(4)\ [Q(4), I(4)] + f(3)\ [I(4) + D(4) - Q(4)]\}$$

1. Korak:

$$0 + 30 - 30 \leq nabava\ (4) \leq 0 + 30$$

$$0 \leq nabava\ (4) \leq 30$$

$$f(4)[0] \rightarrow \min \begin{cases} [0,0] + [0 + 30 - 0] = 0 + 0 \cdot 1{,}5 + 612{,}5 = \mathbf{612{,}5} \\ [15,0] + [0 + 30 - 15] = 250 + 0 \cdot 1{,}5 + 545 = 795 \\ [30,0] + [0 + 30 - 30] = 250 + 0 \cdot 1{,}5 + 522{,}5 = 772{,}5 \end{cases}$$

---

2. Korak:

$$15 + 30 - 30 \leq nabava\ (4) \leq 15 + 30$$

$$15 \leq nabava\ (4) \leq 45$$

$$f(4)[15] \rightarrow \min \begin{cases} [15,15] + [15 + 30 - 15] = 250 + 15 \cdot 1{,}5 + 612{,}5 = 885 \\ [30,15] + [15 + 30 - 30] = 250 + 15 \cdot 1{,}5 + 545 = 817{,}5 \\ [45,15] + [15 + 30 - 45] = 250 + 15 \cdot 1{,}5 + 522{,}5 = \mathbf{795} \end{cases}$$

---

3. Korak:

$$30 + 30 - 30 \leq nabava\ (4) \leq 30 + 30$$

$$30 \leq nabava\ (4) \leq 60$$

$$f(4)[30] \rightarrow \min \begin{cases} [30,30] + [30 + 30 - 30] = 250 + 30 \cdot 1{,}5 + 612{,}5 = 907{,}5 \\ [45,30] + [30 + 30 - 45] = 250 + 30 \cdot 1{,}5 + 545 = 840 \\ [60,30] + [30 + 30 - 60] = 250 + 30 \cdot 1{,}5 + 522{,}5 = \mathbf{817{,}5} \end{cases}$$

Tablica 10: Finalna verzija tablice rješenja stvarnog problema nabave

I	$Q(1)$	$f(1)$	$Q(2)$	$f(2)$	$Q(3)$	$f(3)$	$Q(4)$	$f(4)$
0	30	250	0	272,5	45	522,5	0	612,5
15	45	272,5	0	317,5	60	545	45	795
30	60	295	45	545	60	612,5	60	817,5

(Izvor: izrada autora rada)

Izračun ukupnih troškova nabave:

$$I(i-1) = I(i) + D(i) - Q(i)$$

Tablica 11: Tablica kretanja količina i ukupnih troškova za stvarni problem

Razdoblje (i)	Zalihe $I(i-1)$	Nabava $Q(i)$	Potražnja $D(i)$	Zalihe $I(i)$	Troškovi	
					tr. nabave $Cp(i)$	tr. zaliha $Ch(i)$
1	0	60	30	30	250	45
2	30	0	15	15	-	22,5
3	15	60	45	30	250	45
4	30	0	30	0	-	0
			Ukupno		500	112,5
			Sveukupno		612,5	

(Izvor: izrada autora rada)

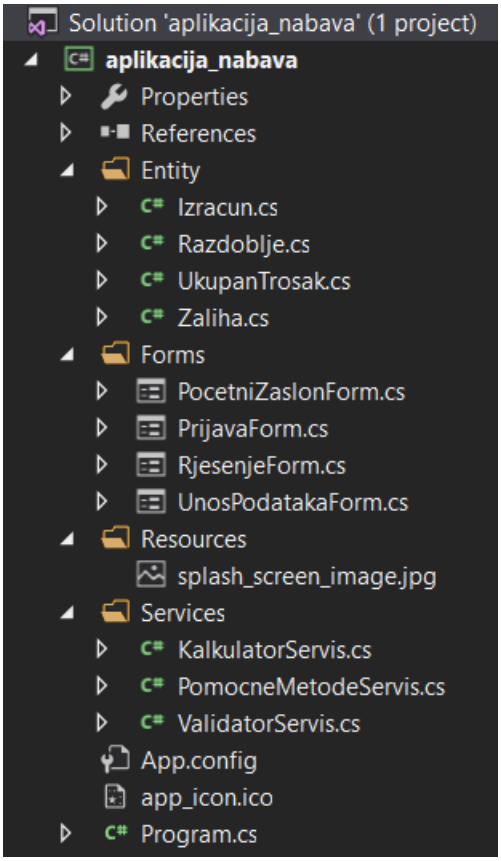
Popunjavanjem tablice kretanja količina i ukupnih troškova dobivaju se ukupni troškovi nabave 500,00 kn i ukupni troškovi zaliha 112,5 kn što zajedno čini 612,5 kn ukupnog troška nabavnog procesa od 4 razdoblja.

7. Programsko rješenje za problem nabave

Programsko rješenje za rješavanje problema nabave korištenjem algoritma dinamičkog programiranja realizirano je u programskom jeziku *C#* u obliku desktop (eng. *Windows* Forms) aplikacije.

7.1. Struktura koda

Na prvoj slici vidljiva je struktura projekta. U direktoriju ***Entity*** nalaze se entiteti odnosno osnovni nositelji podataka koji se koriste kroz aplikaciju. Direktorij ***Forms*** sadrži forme koje se prikazuju kroz aplikaciju. Temeljni dio logike algoritma dinamičkog programiranja za rješavanje problema nabave nalazi se u pomoćnim servisima vidljivima unutar direktorija ***Services.*** U direktoriju ***Resources*** nalazi se slika koja se prikazuje na početnom zaslonu kod učitavanja aplikacije.



Slika 4: Struktura projekta (Izvor: izrada autora rada)

Van navedenih direktorija nalazi se bazna klasa *Program* koja inicijalizira početnu formu i pokreće aplikaciju. U nastavku rada slijedi opis vidljivih klasa te objašnjenje programske logike u istima.

7.2. Opis i objašnjenje koda

Redoslijed kojim će biti objašnjeni pojedini segmenti aplikacije uključuje sljedeći poredak:

1. entiteti nositelji podataka;
2. servisi;
3. forme.

Razlog toga jest hijerarhija odnosa između navedenih segmenata. Preciznije, za razumijevanje logike operacija unutar servisa, potrebno je biti upoznat sa strukturom navedenih entiteta. S druge strane, razumijevanje toka obrade podataka u korisničkim formama uvjetovano je prethodnim upoznavanjem sa radom servisa.

7.2.1. Entiteti

Kako je prethodno spomenuto, entiteti su zamišljeni kao osnovni nositelji odnosno modeli podataka, kako to i biva u objektno orjentiranoj paradigmi. Prvi u nizu jest entitet *Zaliha.*

```
public class Zaliha
{
    public int kolicina { get; set; }
    public double f { get; set; }
    public int Q { get; set; }
}
```

Navedeni entitet čine tri atributa, *kolicina, f i Q.* Atribut *kolicina* predstavlja jednu od mogućih količina u skupu mogućih količina na skladištu, dok atributi *f* i *Q* predstavljaju funkciju uspjeha smanjenja troškova te količinu nabave za dobivenu minimalnu vrijednost odgovarajuće količine na skladištu u određenom koraku rješavanja.

Sljedeći u nizu entiteta jest entitet *Razdoblje*. Svako razdoblje sadrži osnovne atribute poput pripadajućeg rednog broja, količine koja se potražuje te liste zaliha koja se sastoji od mogućih količina zaliha na skladištu.

```
public class Razdoblje
{
    public int redniBrojRazdoblja { get; set; }
    public int potraznja { get; set; }
    public List<Zaliha> listaZaliha { get; set; }
```

```
        // atributi za finalno rješenje
        public int kolicinaZaliha { get; set; }
        public int kolicinaNabave { get; set; }
        public double ukupanTrosakNabave { get; set; } //Cp(i)
        public double ukupanTrosakZaliha { get; set; } //Ch(i)
    }
```

Pored osnovnih atributa, svako razdoblje sadrži i podatke potrebne u tablici kretanja količina i ukupnih troškova. Ti atributi su: *kolicinaZaliha*, *kolicinaNabave*, *ukupanTrosakNabave* i *ukupanTrosakZaliha*. Na osnovu navedenih atributa dobiva se ukupni trošak nabavnog procesa.

Entitet kojeg čine ulazni parametri svakog problema nabave jest entitet *Izracun*.

```
    public class Izracun
    {
        public int maksimalnaKolicinaZaliha { get; set; }
        public int maksimalnaKolicinaNabave { get; set; }
        public double cijenaNabave { get; set; }
        public double trosakZalihaPoJedinici { get; set; }
        public int nabavnaRata { get; set; }
        public int brojPerioda { get; set; }
        public List<int> moguceKolicineZaliha { get; set; }
        public List<Razdoblje> listaRazdoblja { get; set; }
    }
```

Uz osnovne atribute čija se uloga može prepoznati iz njihovih naziva, entitet sadrži i metodu za kreiranje liste razdoblja.

```
 public void kreirajListuRazdoblja(int ukupnoRazdoblja, Form form)
 {
    for (int i = 0; i < ukupnoRazdoblja; i++)
    {
       TextBox potraznjaRazdobljaTextBox =
(TextBox)PomocneMetodeServis.dohvatiKontroluPoImenu(form, "razdoblje" + (i
+ 1) + "TextBox");
       Razdoblje razdoblje = new Razdoblje(i + 1,
int.Parse(potraznjaRazdobljaTextBox.Text));

       foreach (int kolicinaZaliha in this.moguceKolicineZaliha)
       {
         Zaliha zaliha = new Zaliha(kolicinaZaliha);
         razdoblje.listaZaliha.Add(zaliha);
       }
       this.listaRazdoblja.Add(razdoblje);
    }
 }
```

Metoda za ulazni parametar prima ukupan broj razdoblja te unutar *for* petlje kreira svako razdoblje inicijalizacijom objekta klase *Razdoblje*. Zadnja u nisu klasa jest klasa *UkupniTrosak*.

```
public class UkupanTrosak
{
    public double ukupanTrosakNabave { get; set; }
    public double ukupanTrosakZaliha { get; set; }
    public double ukupanTrosakNabavnogProcesa { get; set; }
}
```

Klasa ukupni trošak jednostavne je strukture i služi za pohranu podataka dobivenih u finalnoj tablici kretanja količina i ukupnih troškova.

7.2.2. Servisi

U radu aplikacije nekolicina metoda koristi se na više od jednog mjesta. Kako bi se omogućilo ponovno korištenje spomenutih metoda te održala što bolja struktura koda, iste su izdvojene u pomoćne servise.

7.2.2.1. ValidatorServis

Prvi u nizu takvih servisa jest servis *ValidatorServis* u kojem su definirane ukupno četiri metode: *provjeriPopunjenostInputa*, *validirajVrijednosti*, *isNumeric* te *isInteger*. Forme koje se koriste nazivaju se *WindowsForms* te elementi za unos podataka unutar istih nazivaju se *TextBox*.

```
public static bool provjeriPopunjenostInputa(Form form)
{
   bool ok = true;
   foreach (Control c in form.Controls)
   {
      if (c is TextBox)
      {
         TextBox textBox = c as TextBox;
         if (String.IsNullOrWhiteSpace(textBox.Text))
         {
            ok = false;
         }
       }
    }
    return ok;
}
```

Priložena metoda *provjeriPopunjenostInputa* provjerava da su svi elementi za unos podataka popunjeni te vraća rezultat provjere. Kako su svi ulazni parametri algoritma dinamičkog programiranja brojčane vrijednosti, metoda *validirajVrijednosti* služi za validaciju kojom se utvrđuje da su sve vrijednosti koje je korisnik unio brojčane vrijednosti. Preostale dvije metode jesu *isNumeric* i *isInteger*.

```
public static bool IsNumeric(string s)
{
   double rezultat;
   return double.TryParse(s, out rezultat);
}


public static bool IsInteger(string s)
{
   int rezultat;
   return int.TryParse(s, out rezultat);
}
```

Budući da u programskom jeziku *C#* ne postoje definirane metode za direktnu provjeru da li je određena brojčana vrijednost cjelobrojna ili pak sadrži decimalnu točku, a potreba za korištenjem istih javlja se u nekolicini slučajeva, definirane su navedene metode s ciljem održanja što bolje čitljivosti koda.

7.2.2.2. PomocneMetodeServis

Sljedeći servis *PomocneMetodeServis* sadržava metode za manipulaciju sadržajem u korisničkom sučelju odnosno na formama. Kako je pristup određenim kontrolama formi istovremeno ograničen, a potreban van istih, implementirana je metoda *dohvatiKontroluPoImenu* koja po ulaznom parametru *imeKontrole* pretražuje zadanu formu i vraća željenu kontrolu ukoliko je ista pronađena.

```
public static Control dohvatiKontroluPoImenu(Form form, string imeKontrole)
{
   foreach (Control formControl in form.Controls)
   {
       if (formControl is Panel)
       {
          foreach (Control panelControl in formControl.Controls)
          {
             if (panelControl.Name.Equals(imeKontrole))
             {
                return panelControl;
             }
          }
        }
        if (formControl.Name.Equals(imeKontrole))
        {
           return formControl;
        }
   }
   return null;
}
```

Može se primijetiti kako se unutar funkcije vrši provjera da li se kontrola nalazi unutar panela. *Panel* je vrsta kontrole koja služi za grupiranje elemenata unutar jedne forme, a konkretno u ovoj aplikaciji korišten je za prikaz razdoblja unutar vlastitog prozora, gdje je

potrebno dodavanje zasebnog klizača (eng. *scroll* bar) te zadržavanje dimenzija glavnog prozora pri dodavanju/uklanjanju razdoblja (više o tome vidi poglavlje 7.2.3.).

Na početku pokretanja programa omogućen je unos predviđene potražnje za dva razdoblja. Kako broj razdoblja za koja korisnik želi dobiti optimalni plan nabave nije moguće predvidjeti, implementirana je mogućnost dodavanja ili uklanjanja perioda. Metode *kreirajPoljeZaRazdoblje* i *obrisiPoljeZaRazdoblje* služe upravo za to.

```
public static void kreirajPoljeZaRazdoblje(Form form, Panel
unosRazdobljaPanel)
{
    TextBox prvoRazdoljeTextBox = (TextBox)dohvatiKontroluPoImenu(form,
"razdoblje1TextBox");
    TextBox drugoRazdoljeTextBox = (TextBox)dohvatiKontroluPoImenu(form,
"razdoblje2TextBox");
    Label prvoRazdoljeLabel = (Label)dohvatiKontroluPoImenu(form,
"razdoblje1Label");

    int razmak = drugoRazdoljeTextBox.Top - prvoRazdoljeTextBox.Top;
    int lijevaMargina = prvoRazdoljeTextBox.Left;
    int sirina = prvoRazdoljeTextBox.Width;

    TextBox zadnjeRazdoljeTextBox = (TextBox)dohvatiKontroluPoImenu(form,
"razdoblje" + prebrojiRazdoblja(form) + "TextBox");

     Label novoRazdoljeLabel = new Label();
     novoRazdoljeLabel.Name = "razdoblje" + (prebrojiRazdoblja(form) + 1)
+ "Label";
     novoRazdoljeLabel.Top = zadnjeRazdoljeTextBox.Top + razmak + 3;
     novoRazdoljeLabel.Left = prvoRazdoljeLabel.Right -
prvoRazdoljeLabel.Width;
     novoRazdoljeLabel.Width = prvoRazdoljeLabel.Width;
     novoRazdoljeLabel.Text = (prebrojiRazdoblja(form) + 1) + ".
razdoblje:";

     unosRazdobljaPanel.Controls.Add(novoRazdoljeLabel);

     TextBox noviPeriodTextBox = new TextBox();
     noviPeriodTextBox.Name = "razdoblje" + (prebrojiRazdoblja(form) + 1) +
"TextBox";
     noviPeriodTextBox.Top = zadnjeRazdoljeTextBox.Top + razmak;
     noviPeriodTextBox.Left = lijevaMargina;
     noviPeriodTextBox.Width = sirina;

     unosRazdobljaPanel.Controls.Add(noviPeriodTextBox);
}
```

Pri dodavanju novog razdoblja potrebno je dodati *Label* i *TextBox* za isto, a pozicioniranje istih računa se na osnovu pozicija i dimenzija već postojećeg prvog i drugog razdoblja. Na osnovu razmaka između prva dva razdoblja dobiva se potreban razmak koji mora biti između kontrola svih razdoblja. Tako se kod dodavanja novog razdoblja koristi pomoćna metoda *prebrojiRazdoblja* za izračun broja postojećih razdoblja, kako bi se znalo

koje razdoblje je zadnje po redu. Dobivanjem pozicije zadnjeg razdoblja, novo razdoblje dodaje se ispod istoga za prethodno spomenuti razmak.

```
public static int prebrojiRazdoblja(Form form)
{
    int brojPerioda = 0;
    Regex regex = new Regex(@"(razdoblje)(\d+)(TextBox)$");
    foreach (Control formControl in form.Controls)
    {
       if (formControl is Panel)
       {
          foreach (Control panelControl in formControl.Controls)
          {
             if (panelControl is TextBox &&
regex.IsMatch(panelControl.Name))
             {
                brojPerioda = brojPerioda + 1;
             }
          }
       }
     return brojPerioda;
 }
```

Metoda za brisanje razdoblja puno je jednostavnija jer samo uklanja posljednje razdoblje pri čemu ne treba vršiti nikakav izračun pozicioniranja:

```
public static void obrisiPoljeZaRazdoblje(Form form, Panel
unosRazdobljaPanel)
{
    TextBox periodTextBoxZaObrisati = (TextBox)dohvatiKontroluPoImenu(form,
"razdoblje" + prebrojiRazdoblja(form) + "TextBox");
    Label periodLabelZaObrisati = (Label)dohvatiKontroluPoImenu(form,
"razdoblje" + prebrojiRazdoblja(form) + "Label");

    unosRazdobljaPanel.Controls.Remove(periodTextBoxZaObrisati);
    unosRazdobljaPanel.Controls.Remove(periodLabelZaObrisati);
}
```

U servisu *PomocneMetodeServis* još su preostale dvije metode za generiranje i popunjavanje tablice rješenja problema nabave i tablice kretanja količina i ukupnih troškova. Sadržaj metoda poprilično je opsežan te ne uključuje logiku relevantnu za algoritam dinamičkog programiranja, pa isti nije priložen. Navedene metode pozivaju se na samom kraju aplikacije nakon što se izvrše svi izračuni čime dobiveni podaci postaju spremni za prikaz.

7.2.2.3. KalkulatorServis

KalkulatorServis temeljni je i najopsežniji servis koji sadrži ključne metode za rješavanje problema nabave. Obzirom da je izračun mogućih količina zaliha preduvjet za

rješavanje svih koraka algoritma dinamičkog programiranja, prva metoda *izracunajMoguceKolicineZaliha* poziva se na početku izračuna i prima objekt klase *Izracun* kao ulazni parametar. Koristeći njegove atribute *nabavnaRata i maksimalnaKolicinaZaliha* izračunavaju se moguće količine zaliha na skladištu.

```
public static List<int> izracunajMoguceKolicineZaliha (Izracun izracun)
{
    List<int> moguceKolicineZaliha = new List<int>();
    for (int i = 0; i <= izracun.maksimalnaKolicinaZaliha; i = i +
izracun.nabavnaRata)
    {
        moguceKolicineZaliha.Add(i);
    }
    return moguceKolicineZaliha;
}
```

Sljedeća metoda *izracunajMoguceKolicineNabave* koristi se za dobivanje graničnih vrijednosti nabave, točnije rješenja formule:

$$I(i) + D(i) - maks.kol.zaliha \leq Q(i) \geq I(i) + D(i)$$

```
public static List<int> izracunajMoguceKolicineNabave(int zalihe, int
potraznja, Izracun izracun)
{
    int donjaGranica = zalihe + potraznja -
izracun.maksimalnaKolicinaZaliha;
    int gornjaGranica = zalihe + potraznja;

    if (donjaGranica < 0)
    {
       donjaGranica = 0;
    }
    else if (donjaGranica > izracun.maksimalnaKolicinaNabave)
    {
       donjaGranica = izracun.maksimalnaKolicinaNabave;
    }
    gornjaGranica = gornjaGranica > izracun.maksimalnaKolicinaNabave
       ? izracun.maksimalnaKolicinaNabave
       : gornjaGranica;

    List<int> moguceKolicineNabave = new List<int>();

    for (int i = donjaGranica; i <= gornjaGranica; i = i +
izracun.nabavnaRata)
    {
       moguceKolicineNabave.Add(i);
    }
    return moguceKolicineNabave;
}
```

Kod izračuna graničnih vrijednosti bitno je da dobivene vrijednosti gornje i donje granice nisu manje od 0 ili veće od maksimalne količine nabave.

Nakon izračuna mogućih količina zaliha i mogućih količina nabave za tekuće razdoblje kreće obrada razdoblja odnosno rješavanje koraka za svaku od mogućih količina zaliha. Pri tome se koristi metoda *obradiRazdoblja*.

```
public static void obradiRazdoblja (Izracun izracun)
{
   foreach (Razdoblje razdoblje in izracun.listaRazdoblja)
   {
      if (razdoblje.redniBrojRazdoblja.Equals(1))
      {
          KalkulatorServis.obradiPrvoRazdoblje(izracun, razdoblje);
      }
      else
      {
          KalkulatorServis.obradiRazdoblje(izracun, razdoblje);
      }
   }
}
```

Budući da se postupak rješavanja prvog koraka i ostalih koraka razlikuju, tijekom prolaska kroz listu razdoblja, ovisno o razdoblju poziva se odgovarajuća metoda. Tako se za prvo razdoblje poziva metoda *obradiPrvoRazdoblje*, a za preostala razdoblja metoda *obradiRazdoblje*.

Metoda *obradiPrvoRazdoblje* poziva metodu *izracunajMoguceKolicineNabave* te za zadanu količinu određuje cijenu iste. Ako je količina nabave 0 tada će i cijena nabave biti 0, dok će u protivnom iznositi cijenu nabave zadanu kod unosa podataka od strane korisnika.

```
public static void obradiPrvoRazdoblje(Izracun izracun, Razdoblje
razdoblje)
{
    foreach (Zaliha zaliha in razdoblje.listaZaliha)
    {
       List<int> moguceKolicineNabave =
izracunajMoguceKolicineNabave(izracun.maksimalnaKolicinaZaliha,
razdoblje.potraznja, izracun);

       int index = razdoblje.listaZaliha.IndexOf(zaliha);
       double cijenaNabaveZaKolicinu =
moguceKolicineNabave[index].Equals(0)
        ? 0
        : izracun.cijenaNabave;

       zaliha.f = cijenaNabaveZaKolicinu + zaliha.kolicina *
izracun.trosakZalihaPoJedinici;
       zaliha.Q = moguceKolicineNabave[index];
     }
}
```

Zbrajanjem cijene nabave i troška čuvanja zadane količine zaliha dobiva se vrijednost funkcije uspjeha smanjenja troškova. Dobivena vrijednost pohranjuje se u objekt klase *Zaliha* zajedno sa pripadajućom količinom nabave.

Za ostala razdoblja primjenjuje se postupak definiran u metodi *obradiRazdoblje*, razrađen u prethodnim poglavljima. Za svako razdoblje provodi se N koraka rješavanja u kojima je cilj minimizacija funkcije uspjeha smanjenja troškova.

```
public static void obradiRazdoblje(Izracun izracun, Razdoblje razdoblje)
{
   foreach (Zaliha zaliha in razdoblje.listaZaliha)
   {
        List<int> moguceKolicineNabave =
izracunajMoguceKolicineNabave(zaliha.kolicina, razdoblje.potraznja,
izracun);

        List<double> rjesenjaKoraka = new List<double>();
        foreach (int kolicinaNabave in moguceKolicineNabave)
        {
     // formula algoritma: [nabava, zaliha] + [zalihe + potražnja - nabava]
            int drugeUglateZagradeRezultat = zaliha.kolicina +
razdoblje.potraznja - kolicinaNabave;
            double fPrethodnogRazdoblja =
pronadjiVrijednostIzPrethodnogRazdoblja(drugeUglateZagradeRezultat,
izracun, razdoblje);
            double cijenaNabave = kolicinaNabave.Equals(0) ? 0 :
izracun.cijenaNabave;

            rjesenjaKoraka.Add(cijenaNabave + zaliha.kolicina *
izracun.trosakZalihaPoJedinici + fPrethodnogRazdoblja);
        }

        int indexMinimalneKolicine =
rjesenjaKoraka.IndexOf(rjesenjaKoraka.Min());

        zaliha.f = rjesenjaKoraka.Min();
        zaliha.Q = moguceKolicineNabave[indexMinimalneKolicine];
     }
}
```

Kod izračuna vrijednosti funkcije uspjeha smanjenja troškova, prateći formulu

$$f(i)\,[I(i)] = \min\,\{g(i)\,[Q(i), I(i)] + f(i-1)\,[I(i) + D(i) - Q(i)]\},$$

za svaku od navedenih količina zaliha u drugim uglatim zagradama dobiva se broj po kojem pretražujemo dobivenu vrijednost funkcije za tu količinu zaliha u prethodnom razdoblju.

Pronalazak navedene vrijednosti izdvojen je u zasebnu metodu *pronadjiVrijednostIzPrethodnogRazdoblja*, vidljivu na sljedećoj strani.

```
public static double pronadjiVrijednostIzPrethodnogRazdoblja (int
kolicinaZaliha, Izracun izracun, Razdoblje trenutnoRazdoblje)
{
    int indexTrenutnogRazdoblja =
izracun.listaRazdoblja.IndexOf(trenutnoRazdoblje);
    Razdoblje prethodnoRazdoblje =
izracun.listaRazdoblja[indexTrenutnogRazdoblja - 1];

    foreach (Zaliha zaliha in prethodnoRazdoblje.listaZaliha)
    {
       if (zaliha.kolicina.Equals(kolicinaZaliha))
       {
          return zaliha.f;
       }
    }
    return 0;
}
```

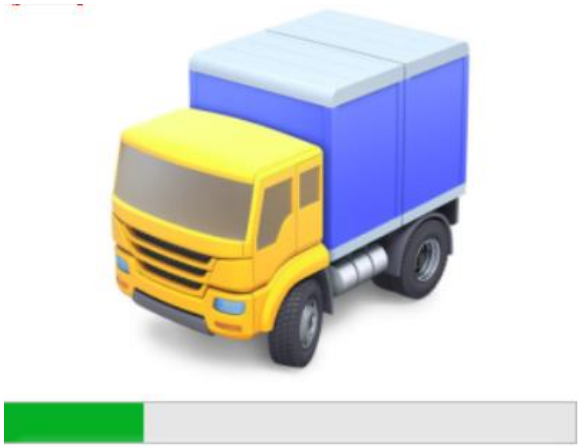
Provedbom provedenih izračuna za dobivanje količine nabave Q i vrijednosti funkcije cilja smanjenja troškova za sva razdoblja, dobivaju se svi podaci potrebni za potpuno popunjavanje tablice rješenja problema nabave. Tada se poziva zadnja u nizu metoda, tj. metoda *izracunajKretanjeKolicinaTijekomRazdoblja*.

```
public static void izracunajKretanjeKolicinaTijekomRazdoblja (Izracun
izracun)
{
  izracun.listaRazdoblja.Reverse();
  foreach (Razdoblje razdoblje in izracun.listaRazdoblja)
  {
     if (razdoblje.redniBrojRazdoblja.Equals(izracun.listaRazdoblja.Count))
     {
        razdoblje.kolicinaZaliha = 0;
     }
     foreach (Zaliha zaliha in razdoblje.listaZaliha)
     {
         if (zaliha.kolicina.Equals(razdoblje.kolicinaZaliha))
         {
            razdoblje.kolicinaNabave = zaliha.Q;
         }
     }
     if (!razdoblje.redniBrojRazdoblja.Equals(1))
     {
         int indexTrenutnogRazdoblja =
izracun.listaRazdoblja.IndexOf(razdoblje);
         Razdoblje prethodnoRazdoblje =
izracun.listaRazdoblja[indexTrenutnogRazdoblja + 1];
         //formula -> I(i-1) = I(i) + D(i) - Q(i)
         prethodnoRazdoblje.kolicinaZaliha = razdoblje.kolicinaZaliha +
razdoblje.potraznja - razdoblje.kolicinaNabave;
      }
   }
   izracun.listaRazdoblja.Reverse();
}
```

Kao što samo ime metode sugerira, metoda izračunava kretanje količina tijekom razdoblja te priprema vrijednosti za popunjavanje tablice kretanja zaliha i ukupnih troškova.

7.2.3. Korisničke forme

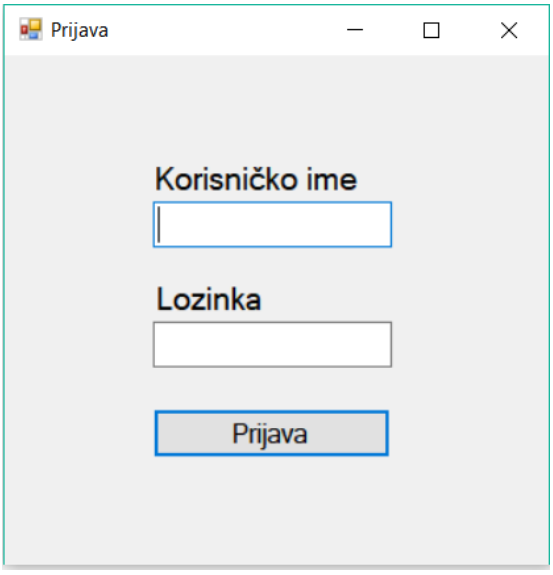
Dio koda zaslužan za vizualni prikaz aplikacija nalazi se u formama. Aplikacija sadrži ukupno četiri forme odnosno korisnička zaslona. Prvi korisnički zaslon *PocetniZaslonForm* prikazuje se za vrijeme učitavanja aplikacije.



Slika 5: Početni zaslon (Izvor: izrada autora rada)

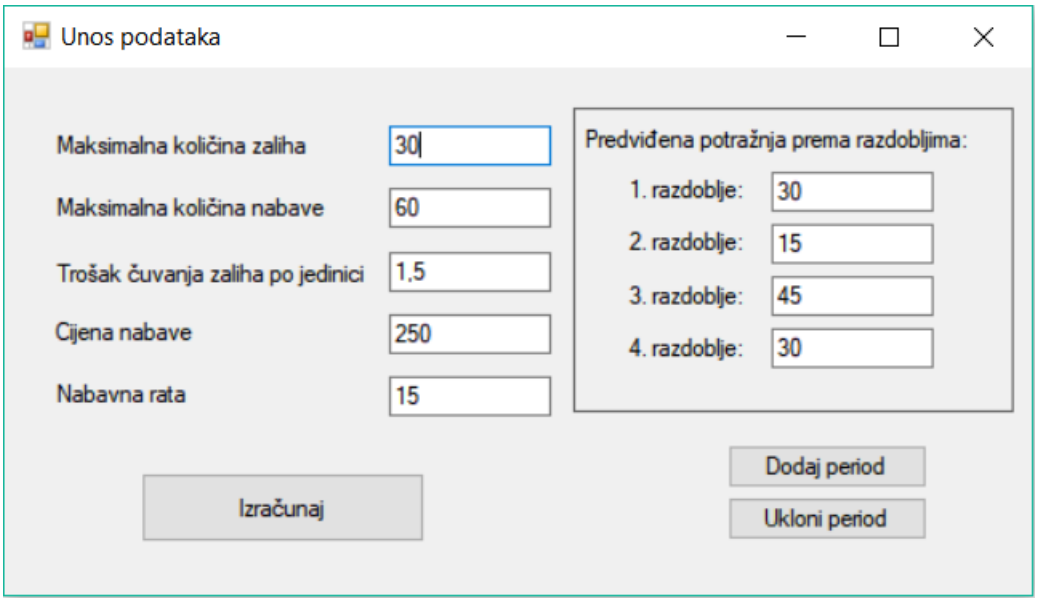
Nakon uspješnog učitavanja aplikacije korisnik je preusmjeren na zaslon za prijavu u aplikaciju. Korisnički podaci za prijavu su:

- Korisničko ime – admin;
- Lozinka – admin.



Slika 6: Zaslon za prijavu (Izvor: izrada autora rada)

U slučaju da se unesu krivi podaci, ispisuje se poruka o pogrešci. Ukoliko je korisnik unio ispravne podatke, otvara se zaslon za unos ulaznih parametara za problem nabave.

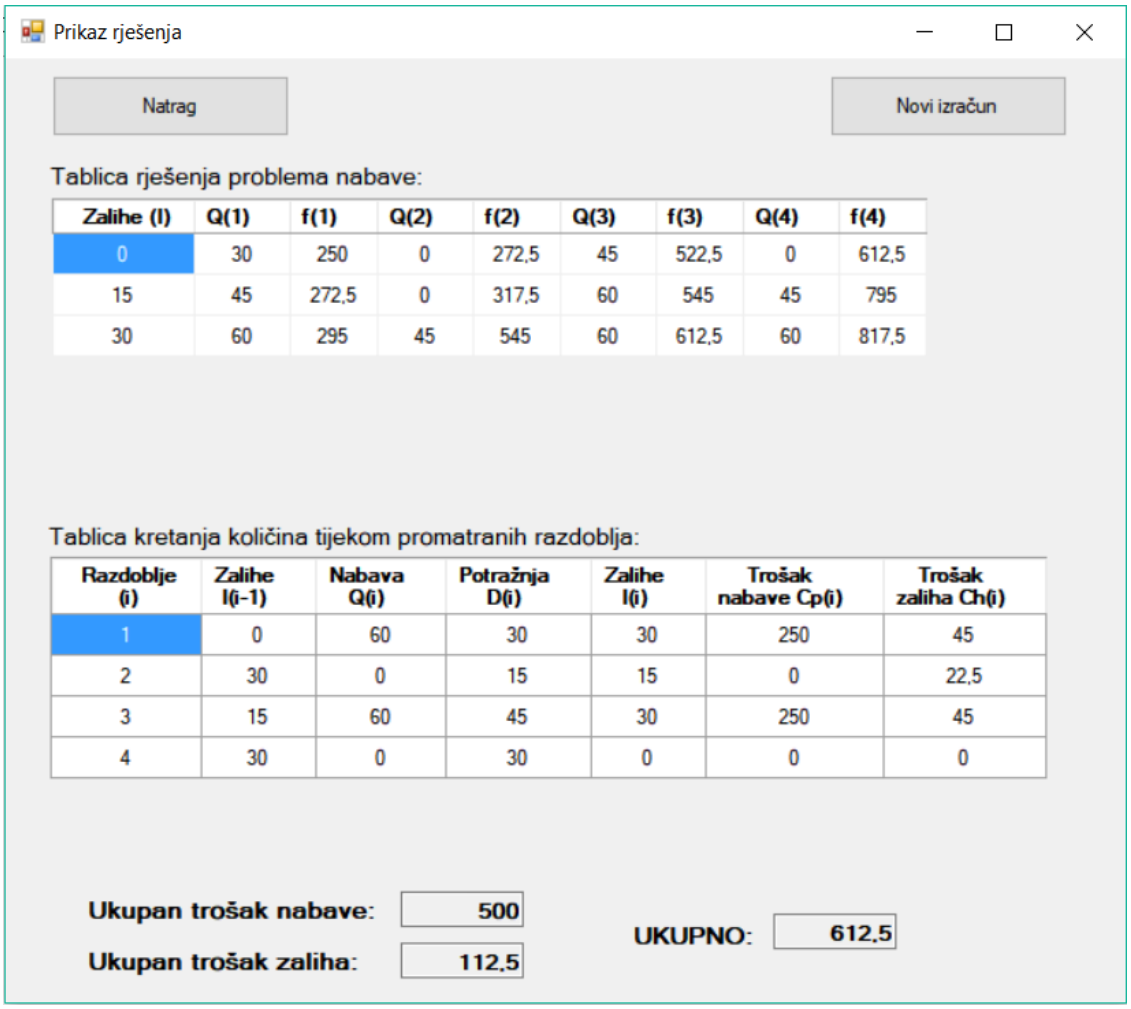


Slika 7: Zaslon za unos podataka (Izvor: izrada autora rada)

Kao što se da primijetiti, uneseni su podaci stvarnog problema obrađenog u prethodnom poglavlju. Korisnik ima mogućnost birati broj razdoblja za koje želi rješavati problem nabave. Minimalni broj razdoblja ograničen je na 2 jer u protivnom problem nabave kao višeetapnog procesa ne bi imao smisla. Klikom na gumb *Izračunaj* provjerava se ispravnost unesenih podataka te u slučaju uspješne validacije, korisniku se prikazuje rješenje. Ograničenja koja se provjeravaju su:

- Sva polja za unos moraju biti popunjena;
- Sva polja za unos moraju biti brojčane vrijednosti;
- Svi podaci osim troška čuvanja zaliha po jedinici i cijene nabave moraju biti cjelobrojne vrijednosti.

Zaslon sa rješenjem prikazan je na sljedećoj stranici.



Slika 8: Zaslon za prikaz rješenja (Izvor: izrada autora rada)

Nakon dobivanja rješenja problema nabave korisnik ima mogućnost povratka na prethodni zaslon unosa podataka i mogućnost novog izračuna. Ukoliko korisnik klikne na gumb *Natrag*, podaci u poljima za unos će biti sačuvani, dok se klikom na gumb *Novi izračun* također vraća na prethodni zaslon, no tada su polja za unos prazna.

8. Zaključak

Sve brži razvoj tehnologije uvelike utječe na rast poduzeća i dinamike razmjene dobara na tržištu, stoga se povećava i potreba za asimilacijom poduzeća novim trendovima i zahtjevima iz okoline. Razmjenu dobara čine procesi nabave i prodaje, kao jedni od najbitnijih segmenata odnosno procesa svake organizacije. Bilo da se radi o nabavi materijalnih resursa ili onih nematerijalnih, optimizacija procesa nabave može uvelike utjecati na pozitivan odnos prihoda i rashoda pojedine organizacije, što je na kraju krajeva jedan od glavnih pokazatelja uspješnog poslovanja istih.

Prije kretanja u sam proces rješavanja problema nabave odnosno minimizacije troškova iste, potrebno je razumijeti pojam nabave kao višeetapnog procesa i zaliha, obzirom da su iste neizostavan dio svakog nabavnog procesa.

Pravovremeno i pouzdano planiranje nabave jedan je od krucijalnih preduvjeta minimizacije odnosno uklanjanja nepotrebnih troškova nastalih tijekom procesa nabave. Metoda dinamičkog programiranja jedan je od alata koji olakšava planiranje nabave kao višeetapnog procesa te pruža detaljan uvid u stanje zaliha te količine nabave svakog razdoblja za koje se provodi izračun troškova.

Primarni cilj ovog rada bio je prikazati stvarni problem nabave te primijeniti metodu dinamičkog programiranja za rješavanje istog, što je uspješno provedeno te je dobiven uvid u kretanje količina i ukupnih troškova tijekom danih razdoblja. Takav uvid poduzeću pruža mogućnost bolje raspodjele troškova, uviđanje potencijalnih neočekivanih troškova i pravovremeno reagiranje u slučaju nastanka problema.

Definiranjem i objašnjenjem algoritma dinamičkog programiranja i problema nabave ovaj rad pruža temeljitu teoretsku osnovu za razumijevanje istih. Kao dodatak teoretskom dijelu, rad uključuje i praktični dio u kojem je izrađeno programsko rješenje. Programsko rješenje realizirano je u obliku samostojeće (eng. *Desktop*) aplikacije koja omogućuje bržu praktičnu primjenu metode dinamičkog programiranja za rješavanje problema nabave u stvarnim situacijama.

9. Literatura

[1] Ammer, C., Ammer, D.S., Dictionary of Business and Economics, The Free Press, London, 1984.

[2] Dobrenić, S., Operativno istraživanje, Fakultet organizacije i informatike, Varaždin, 1978.

[3] Ferišak, V., Nabava: politika, strategija, organizacija, management, 2. aktualizirano i dopunjeno izdanje, vlastita naklada., Zagreb, 2006.

[4] Ferišak, V., Stihović, L., Nabava i materijalno poslovanje, Informator, Zagreb, 1989.

[5] Jovan Petrić, Zora Petrić, Kojić, Šarenac, Operaciona istraživanja u vojsci, Vojnoizdavački zavod Beograd, Beograd, 1974.

[6] Knežević, B., Količina zaliha kao čimbenik menadžmenta nabave, Ekonomski fakultet, Zagreb, 2012.

[7] Krpan, Maršanić, Jedvaj, Upravljanje zalihama materijalnih dobara i skladišno poslovanje u logističkoj industriji, https://hrcak.srce.hr/129385**,** Preuzeto: 26. svibnja, 2019.

[8] Majstorović, V., Upravljanje proizvodnjom i projektima, Sveučilište u Mostaru, Mostar, 2001.

[9] Prester, J., Upravljanje lancima dobave, Sinergija nakladništvo, Zagreb, 2012.

[10] Q Alliance, Q promotivna brošura, Zagreb, 2018.

[11] Techie Delight, Dynamic Programming Practice Problems, https://www.techiedelight.com/Category/dynamic-programming/, Dostupno: 25. lipnja 2019,

[12] Wiendahl, H. P., Load – Oriented Manufacturing Control, Springer Verlag, Berlin, Heidelberg, New York, 1995.

[13] Winston, W. L., Operation Research Applications and Algorithms, Duxbury Press, Belmont, California, 1994.

[14] Žilbert, B, Strateška nabava, Mate d.o.o., Zagreb, 2007.

10. Popis slika

11. Popis tablica