Posebna svojstva simpleks algoritma

Mašek, Maja

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

Maja Mašek

POSEBNA SVOJSTVA SIMPLEKS ALGORITMA

ZAVRŠNI RAD

Varaždin, 2018.

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

Maja Mašek
Matični broj: 41343/12-I
Studij: Poslovni sustavi

POSEBNA SVOJSTVA SIMPLEKS ALGORITMA

ZAVRŠNI RAD

Mentor:
Dr. sc. Nenad Perši

Varaždin, veljača 2018.

Sadržaj

1. Uvod

Operacijska istraživanja imaju široko matematičko, ali i ekonomsko poimanje. Ona se sastoje od različitih metoda i tehnika koja svojom primjenom uveliko potpomažu ekonomskim analizama odnosno donošenju krucijalnih odluka unutar nekog poduzeća. Područje operacijskih istraživanja koje će se obrađivati u ovome radu je linearno programiranje. Svakako prvotna uporaba i nastanak operacijskih istraživanja odnosno linearnog programiranja nije bio namijenjen poduzećima kakva danas postoje, ali sam razvoj discipline pratio je drugi svjetski rat i povećan razvoj industrije nakon rata odnosno razvoj računala. Računalo je stupilo na snagu unutar discipline kao vrlo bitan element pomoću kojeg se točnije i brže dolazi do rješenja problema zadanog sustava. Načini odnosno metode pomoću kojih se dolazi do takvih rješenje je mnogo. Jedna od važnijih obradit će se u ovome radu, a to je simpleks metoda sa svojim algoritmom, koja kao metoda spada pod metodu linearnog programiranja. Osim osnovnog o ovim metodama bit će riječi i o posebnim slučajevima koji se mogu javiti prilikom primjene simpleks algoritma. Ono što se ovim radom želi pokazati je zašto se ti slučajevi javljaju i na koji način im se pristupa. U realnim sustavima u kojima se radi s velikim brojem informacija i podataka vrlo je bitno imati i matematičko uporište u donošenju odluka. Zato se ove metode jako dobro mogu prilagoditi takvim oblicima problema u kojima treba dobro razvrstati podatke i primijeniti ih na odgovarajući način. Naravno, uvijek kada se radi o velikom broju podataka koje treba obraditi odnosno na temelju kojih treba doći do kvalitativnih odgovora, prisutna je mogućnost pojave određenih poteškoća. Poteškoće mogu biti dio krivo postavljenih problema, zato ih je bitno prepoznati i uvidjeti uzroke takvih poteškoća. Zato će ovaj rad obraditi upravo te važne dijelove simpleks algoritma. Nakon same definicije i objašnjenja linearnog programiranja te zatim i detaljnog opisa funkcioniranja simpleks algoritma odgovorit će se na pitanje zašto se određene poteškoće prilikom primjene tog algoritma javljaju i mogu li se na neki način i otkloniti kako se ne bi dobila višesmislena rješenja problema. Tema ovog rada vrlo je kompleksna i zahtjevna, ali ona će biti odrađena kroz što jednostavnije primjere i definicije kako bi se shvatila cjelokupna bit.

2. Operacijska istraživanja

Postoje mnoga tumačenja i definicije operacijskih istraživanja. Može se reći kako operacijska istraživanja objedinjuju matematiku, programiranje, elektroničku obradu podataka, organizaciju podataka, pribavljanje podataka, tehnike strukturiranja i klasificiranja, kreativne tehnike i još mnogo disciplina i područja s kojima operacijsko istraživanje mora 'surađivati' kako bi uspjelo postići svoj cilj i primjenu. [Barković, 2001, 8]. Ovoliko disciplina koje su zajedno ujedinjene prema jednom cilju uvodi se kako bi se učvrstilo prihvaćanje odluka na znanstvenoj osnovi. Operativno istraživanje kao znanstvena metoda eksplicitno se pojavila pod tim imenom za vrijeme drugog svjetskog rata na području vojne strategije i taktike, a kasnije se zbog svoje djelotvornosti proširilo i na druga područja ljudske djelatnosti [Martić, Vandal, 1968, 3]. Operacijska istraživanja objedinjuju mnoge metode i matematička i statistička sredstva uz pomoću kojih se zajedno dolazi do rješenja problema. Na temelju klasifikacije problema određuje se koje će se od metoda i sredstava primijeniti za rješavanje problema, ali i njegove analize. Za kvalifikaciju problema koriste se dva klasifikacijska kriterija [Martić, Vandal, 1968, 7]. Prema prvom kriteriju dobiva se klasifikacija prema upotrijebljenim metodama i po tom kriteriju modeli operacijskih istraživanja mogu se klasificirati ovako: modeli linearnog programiranja, modeli kvadratnog programiranja, modeli bilinearnog programiranja itd. [Martić, Vandal, 1968, 8]. Po drugom pak kriteriju dobiva se klasifikacija prema obrađivanom sadržaju i po tom kriteriju modeli se klasificiraju ovako: modeli proizvodnih procesa, modeli transporta, modeli zaliha itd. [Martić, Vandal, 1968, 8]. Modeli koji će se u ovom radu obrađivati i na temelju kojih će se prikazati tema ovog rada su modeli linearnog programiranja, koji će biti prikazani i kao problemi proizvodnih procesa.

3. Linearno programiranje

Linearno programiranje dio je matematičke discipline koja se naziva matematičko programiranje [Kalpić, Mornar, 1996, 5]. Također, linearno programiranje može se svrstati i unutar discipline matematičkog optimiranja. Ono je najprimjenjivanija metoda operativnog istraživanja te se primjenjuje kada se traže ekstremne vrijednosti problema, kao naprimjer maksimum prihoda odnosno minimum troškova. [Dobrenić, 1975, 3]. Može se reći da je linearno programiranje zapravo skup metoda i postupaka kojima se određuju ekstremne vrijednosti takve linearne funkcije, čije područje definicije određuje sustav linearnih jednadžbi ili nejednadžbi [Dobrenić, 1975, 28]. Upravo zato što ova metoda rješava probleme koji su postavljeni tako da su im sva ograničenja i ciljevi odnosno sve jednadžbe i nejednadžbe u linearnom obliku, opravdava naziv same metode. Nadalje, neće se dublje ulaziti u povijest nastanka tehnike linearnog programiranja, ali valja napomenuti neke bitne točke njezina razvoja.

Iako je linearno programiranje kakvo danas postoji dobilo naziv i matematičku definiciju tek kasnih četrdesetih i ranih pedesetih godina 20. stoljeća, njegovom otkrivanju prethodila su razna druga stvaranja, razvoji i utjecaji. Neka čak sežu i u 18. stoljeće za koje se kaže da je bio začetak linearnog programiranja jer su tada ekonomisti prvi puta počeli opisivati ekonomske sisteme matematičkim izrazima [Dobrenić, 1975, 4]. Stoljeće kasnije još jedan je ekonomist dao svoj doprinos razvoju linearnog programiranja time što je preporučio jedan umjetni matematički model koji je kao dio svoje strukture imao tehnološke koeficijente [Dobrenić, 1975, 4]. Od tada pa sve do tridesetih godina 20. stoljeća nije se ništa bitno više događalo na tom polju da bi se prije i nakon drugog svjetskog rata ova teorija vrlo brzo razvila i pretvorila u disciplinu. Bitno ime koje se spominje je ruski matematičar L.V. Kantorovič koji je svojim djelima odnosno radovima približio matematiku ekonomskom planiranju proizvodnje odnosno ekonomskom računanju optimalnog iskorištenja strojeva – kako se i zvalo jedno od njegovih radova [Dobrenić,1975, 5]. Nažalost, ruski matematičar je sa svojim radovima ostao poprilično nezapažen neko vrijeme od objave pa se za to vrijeme već naveliko razvilo linearno programiranje kao disciplina izvan granica SSSR-a. Tako je 1947. Georg. B. Dantzig razradio simpleks računsku metodu za pronalaženje optimalnog mogućeg programa problema linearnog programiranja [Dobrenić, 1975, 6]. O toj metodi odnosno njezinom algoritmu svakako će biti više riječi u narednim poglavljima. Nakon primjene linearnog programiranja u vojnom području - planiranja vojnih akcija širokih razmjera, od ruta brodova između luka do utvrđivanja tijeka dobara između privrednih grana – slijede prvi pokušaji

primjene linearnog programiranja u rješavanju problema industrije [Barković, 2001, 10]. Danas se već mogu navesti mnogi problemi koji se rješavaju uz pomoć linearnog programiranja poput proizvodnih programa, izbora lokacije tvornica, optimalnog planiranja investicijskih ulaganja, razmještaja strojeva, izbor optimalnih tehnoloških postupaka, sastavljanja optimalnih planova prehrane, transporta, izbora i razmještaja sredstava naoružanja i još bezbroj drugih područja na kojima je moguće primijeniti linearno programiranje.

Trebalo bi napomenuti kako je linearno programiranje doživjelo 'procvat' odnosno povećan interes za korištenjem paralelno s razvojem računala i računalnih tehnika kojima se omogućilo brže rješavanje sve većih zahtjeva i problema koji nisu bili mogući samo uz pomoć ljudskog faktora. Odlučivanje u organizaciji moralo je biti poduprto i računalnim programima uz pomoć kojih su se predlagali precizni i točni podaci. Svakako da su se ti programi iz godinu u godinu usavršavali i nadograđivali, pa se tako još i danas oni mijenjaju i pokušavaju što više prilagoditi željenim problemima. Danas se tako uz korištenje računala rješavaju vrlo složeni problemi optimizacije koji se javljaju unutar područja u kojima je linearno programiranje nezaobilazna metoda [Kuzmanović,Sabo, 1]. Problemi transporta, energije, telekomunikacije, proizvodnje – svi oni nedvojbeno moraju uključiti u svoje poslovanje odlučivanje na temelju rezultata linearno programiranja.

Prema izvoru [Ravindran, Phillips, Solberg, 2007, 13] pojam linearnog programiranja jednostavno definiramo kao određen skup programskih problema koji zadovoljavaju sljedeće uvjete:

1. Varijable odlučivanja koje su uključene u problem su *nenegativne* odnosno pozitivne ili nula. To se može objasniti tako da ako varijable gledamo kao proizvod on se može proizvoditi u nekoj količini ili se ne proizvodi, ne može se negativno proizvoditi.
2. Kriterij za odabir najbolje vrijednosti varijabli odlučivanja može biti opisan linearnom funkcijom tih varijabli. Funkcija kriterija obično se naziva *funkcija cilja.*
3. Operativna pravila koja uređuju proces npr. količina resursa mogu se izraziti kao skup linearnih jednadžbi ili linearnih nejednadžbi. Ovaj skup se obično naziva *skup ograničenja*.

Upravo su zadnja dva stanja razlozi za zašto se *linearno programiranje* tako i naziva. Kako se već i navelo oni su u linearnom obliku. Široka uporaba linearnog programiranja seže od već spomenute vojske, preko ekonomije do industrije pa čak i socijalnih problema. Navode se tri razloga za njihovo široko korištenje:

1. Raznoliki problemi iz različitih područja mogu se prikazati ili barem približno kao model linearnog programiranja.
2. Na raspolaganju postoje odnosno dostupne su učinkovite tehnike za riješevanje problema linearnog programiranja.
3. Sveprisutna jednostavnost kojom se različiti podaci mogu upravljati putem modela linearnog programiranja.

Može se još naglasiti kako su procedure rješavanja po prirodi iterativne i da se čak i za umjereno velike probleme pribjegava korištenju računala kako bi se došlo do rješenja. Ponekad su analize do kojih se došlo linearnim programiranjem bile neisplative jer je trošak procedure s kojom se došlo do rješenja problema bio veći nego vrijednost samog rješenja. Danas to naravno nije takav slučaj jer je razvoj računala odnosno i same tehnologije toliko uznapredovao da se je rješenje velikih problema linearnog programiranja postalo ne samo izvedimo već i jeftino [Ravindran, Phillips, Solberg, 2007, 14].

Metode linearnog programiranja najvažniji su instrument operacijskih istraživanja [Barković, 2001, 9]. Kako operacijska istraživanja imaju za cilj odrediti najbolji odnosno optimalni smjer aktivnosti u problemu odlučivanja u okviru danih restrikcija i ograničenja kapaciteta [Barković, 2001, 1], tako se može reći da je svrha linearnog programiranja unutar operacijskih istraživanja također optimiranje. Takav cilj onda promatramo čisto sa stajališta ekonomskog problema, a ne matematičkog, iako je ono potrebno kako bi se problem uspio riješiti odnosno optimirati. Upravo takvo matematičko definiranje linearnog programiranja glasi, kako smo već naveli, da je ono rješenje nekog matematičkog zadatka koji se sastoji u optimiranju neke linearne funkcije čije varijable zadovoljavaju neki sustav linearnih jednadžbi i nejednadžbi [Barković, 2001, 9]. Striktno govoreći kako je već spomenuto linearno programiranje i je matematička tehnika, ali u ekonomici nekog poduzeća linearno programiranje obuhvaća i više nego poznavanje same tehnike [Dobrenić,1966, 25]. Linearno programiranje je skup metoda čija je karakteristika univerzalnost u primjeni pa one nisu ograničene na usko područje ekonomskih analiza. S druge pak strane matematička

formulacija i postupak omogućavaju da se dobije točno rješenje, lišeno sumnje i oslobođeno subjektivizma, što je u ekonomskoj analizi izrazito bitno [Dobrenić, 1975, 28].

3.1. Osnovni pojmovi i njihovo definiranje

Prije samog matematičkog definiranja problema linearnog programiranja valjalo bi objasniti osnovne pojmove s kojima se započinje rješavanje problema linearnog programiranja. Neki su se već u kratko i spomenuli. Prvi svakako kojeg treba objasniti je model.

Model određenog problema postavlja se u apstraktnom ili matematičkom obliku. Naravno kako bi se nad modelom koji je apstraktno postavljen mogle primjenjivati određene operacije odnosno metode i tehnike mora ga se dovesti u matematički oblik. Prilikom tog 'prevođenja' odnosno još prilikom same formulacije problema može doći do kritičnih pogrešaka. Ukoliko se izostavi samo jedna činjenica tj. varijabla ili uvjeti koji su od kritične važnosti može se desiti da se pojave rješenja koja nisu relevantna za aktualnu situaciju [Dobrenić, 1966, 14].

Postavljeni model linearnog programiranja sastoji se od kriterija odnosno cilja, ograničenja i uvjeta. Funkcija cilja kako smo već i spomenuli nam služi za izbor najboljeg mogućeg rješenja postavljenog problema odnosno optimalnog rješenja. Zato se ona naziva i funkcijom kriterija. [Vučković, 1983, 14]. Ograničenja pak ograničavaju niz rješenja za vrijednosti programa, ali tek s funkcijom cilja tvore potpuni program. Ograničenja problema postavljaju se u obliku jednadžbi i/ili nejednadžbi.

Na temelju sljedećeg jednostavnog primjera[1] objasnit će se navedeno. Recimo da u zamišljenoj industriji imamo jedan stroj čiji je kapacitet 30 radnih sati tjedno. Želi se odrediti količina proizvoda koja se može obraditi na tom stroju, a koju označavamo s nepoznanicom $\boldsymbol{x}$. Svaka jedinica proizvoda x zahtijeva vrijeme obrade od 5 sati rada. Izraz koji slijedi i ovoga je:

$$5x = 30.$$

Znak jednakosti određuje da se kapacitet stroja mora iskoristiti u potpunosti te na temelju toga dolazi se do sljedećeg rješenja postavljenog problema, a to je $x = 5$. To rješenje naziva se program. Također za ovaj problem moglo se tražiti potpuno iskorištenje, ali da ono nije zahtijevano pa je onda potrebno u tom slučaju koristiti znak nejednakosti „manje od ili

[1] Primjer izrađen prema Dobrenić, 1966, 13.

jednako“ - $\leq$, koji se još može nazivati i ograničenjem koje je zadano gornjom granicom. Ukoliko se u interpretaciji problema zahtijeva iskorištenje minimalno 30 radnih sati odnosno ograničenje je zadano donjom granicom, znak nejednakosti koji se mora staviti je „veće od ili jednako“ - $\geq$. Ne treba bitno naglašavati da nejednadžbe dozvoljavaju niz rješenja odnosno selekciju programa iz širokog niza alternativa, dok jednadžba naravno ne. Ipak, kod jednadžbe je moguće da se pojavi niz rješenja ukoliko recimo na ovom primjeru želimo da se na stroju obavlja izrada još jednog proizvoda i to po jedinici od 2 sata rada na istom. Na temelju toga dobiva se sljedeći oblik jednadžbe:

$$5x_1 + 3x_2 = 30,$$

i njeno rješenje odnosno program koji ju zadovoljava

program 1: $x_1 = 6, \quad x_2 = 0$

program 2: $x_1 = 0, \quad x_2 = 10$

program 3: $x_1 = 3, \quad x_2 = 5.$

Rješenje ove jednadžbe dopušta obradu različitih količina prvog i/ili drugog proizvoda, ali na način da ispunjavaju uvjet koji nam zadaje da program mora sadržavati onolike količine koje zajedno iskorištavaju pun kapacitet stroja. Navedenu jednadžbu s dvije nepoznanice moglo se također zapisati i u obliku nejednadžbe i time bi se dobilo otvoreniji problem u smislu da kapacitet stroja ne bi morao biti iskorišten u potpunosti:

$$5x_1 + 3x_2 \leq 30.$$

Ponekad je prilikom postavljanja problema ključno upravo ovo. Prepoznati odgovarajuće ograničenje nad raspoloživim kapacitetom odnosno prepoznati hoće li se upotrijebiti nejednakost ili jednakost. Ovom primjeru dodat će se još jedno ograničenje tj. još jedan stroj na kojem se dorađuju proizvodi, ali ovog puta samo proizvod x_1 koji zahtijeva 2 sata obrade i to unutar kapaciteta od 12 radnih sata:

$$4x_1 + 0x_2 \leq 12.$$

Ova ograničenja moraju se spojiti u jedan prošireni model jer oba izraza moraju biti simultano zadovoljena. Također potrebno im je dodati uvjet koji ne dozvoljava negativne vrijednosti za proizvode pošto se logično i realno proizvodi ili ne proizvode (vrijednost im je jednaka 0) ili proizvode (vrijednost im je veća od 0). Ne moguće ih je negativno proizvoditi. Takav uvjet naziva se uvjet nenegativnosti. Svako moguće rješenje problema je odgovarajuća kombinacija nenenegativnih vrijednosti svih polaznih varijabli kod koje lijeva strana svake od

ograničavajućih nejednadžbi ne smije imati vrijednost veću od njene odgovarajuće desne strane [Vučković, 1983, 51]. Krajnji oblik modela, ali ne i potpuni sada glasi:

$$5x_1 + 3x_2 \leq 30$$
$$4x_1 + 0x_2 \leq 12$$
$$x_1, x_2 \geq 0.$$

Kako se pomoću postavljenih ograničenja dobiva niz mogućih rješenja, postavlja se pitanje koji od tih programa izabrati kao konačno, najbolje rješenje. U toj odluci pomaže funkcija cilja tj. funkcija kriterija. Svaki problem linearnog programiranja je rješiv ako ima bar jedno moguće rješenje, a isti može imati i više mogućih rješenja. Upravo funkcija cilja služi kao kriterij za izbor najboljeg mogućeg rješenja problema [Vučković, 1983, 48]. Funkcija cilja određuje tijek programa odnosno pomoću nje se određuje hoće li se pomoću programa maksimizirati ili minimizirati konačna vrijednosti. U svakom slučaju za cilj joj je izvršiti optimiranje. Pronaći će najbolje moguće rješenje. Funkcija cilja izražava se pomoću zadanih konstanti npr. u navedenom slučaju $\boldsymbol{c_1}$ i $\boldsymbol{c_2}$ koje mogu prikazivati prihod po jedinici proizvoda ili pak rashod - u ovisnosti želi li se maksimizirati ili minimizirati zadani problem**:**

$$Z = c_1x_1 + c_2x_2 \ .$$

Za navedeni primjer uzet će se vrijednosti:

$$c_1 = 8, \quad c_2 = 3,$$

pa zapis za ukupni prihod izgleda ovako:

$$Z = 8x_1 + 3x_2.$$

Odnosno potpuni model za ovaj problem glasi:

maksimizirati $\quad 8x_1 + 3x_2,$

uz uvjete

$$5x_1 + 3x_2 \leq 30$$
$$4x_1 + 0x_2 \leq 12$$
$$x_1, x_2 \geq 0.$$

Program koji zadovoljava ove uvijete i cilj maksimizacije je: $x_1 = 3, \ \ x_2 = 5$. Ne postoji više niti jedan program koji bi bio tako dobar da rezultira najvećim prihodom uz postavljena ograničenja. Taj maksimalni prihod iznosi $8 \cdot 3 + 3 \cdot 5 = 39$ n.j. kao novčanih jedinica pošto se ne zna i nije trenutno bitno u kojem obliku su one izražene.

Preko ovog malog i jednostavnog primjera željelo se u kratko prikazati na koji način se postavlja model linearnog programiranja i na koji način se ono optimira. Naravno da se u stvarnom poslovnom svijetu pojavljuju daleko veći i kompleksiji modeli i njima odgovarajuće metode. O nekima od njih će kasnije biti više riječi.

Sve ovo navedeno može se zapisati naravno pomoću matematičkih simbola odnosno model linearnog programiranja izgleda tako da problem kojeg želimo optimirati se zapiše kao [Dobrenić, 1966, 17]:

$$\boldsymbol{opt.} \sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{c_j x_j} \,,$$

s njegovim pripadajućim uvjetima

$$\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{a_{ij} x_j} \begin{matrix} \leq \\ \geq \end{matrix} \boldsymbol{b_i} \qquad \boldsymbol{j} = 1, 2, \ldots, \boldsymbol{n} \,,$$

$$\boldsymbol{x_j \geq 0} \qquad \boldsymbol{i} = 1, 2, \ldots, \boldsymbol{m}.$$

Ako se detaljnije razloži funkcija cilja onda poprima sljedeći oblik:

$$\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{c_j x_j} \equiv c_1 x_1 + c_2 x_2 + \ldots + \boldsymbol{c_n x_n} \,,$$

i s njenim ograničenjima

$$\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{a_{ij} x_j} \equiv a_{i1} x_1 + a_{i2} x_2 + \ldots + \boldsymbol{a_{1n} x_n} \begin{matrix} \leq \\ \geq \end{matrix} \boldsymbol{b_i} \,,$$

$$x_j \geq 0 \,,$$

gdje su svakako opet

$$j = 1, 2, \ldots, n \,,$$

$$i = 1, 2, \ldots, m \,.$$

Iz svih navedenih izraza sljedeća tablica prikazat će osnovne pojmove koji će nam biti polazište za razumijevanje ostatka rada u kojemu će se oni bolje objasniti i detaljnije razraditi ponajviše uz odabrane primjere.

Tablica 1. Osnovni pojmovi [Izvori: Dobrenić, 1966 i Babić, 2010]

Funkcija cilja ili kriterija (prihodi ili rashodi)	$opt. \sum_{j=1}^{n} c_j x_j$
Ograničenja na varijable x_j - skup ograničenja	$\sum_{j=1}^{n} a_{ij} x_j \leq \geq b_i$
Funkcija cilja	Z
Strukturne varijable (proizvodi ili komponente)	x_j
Tehnički koeficijenti ili normativi	a_{ij}
Uvjeti nenegativnosti	$x_j \geq 0$
Cijena j-tog proizvoda ili komponente	c_j
Kapaciteti ili resursi	b_i
Broj varijabli (proizvoda ili komponenata)	n
Broj ograničenja	m

Može se zaključiti kako su jedine konstante koje su zadane u problemu linearnog programiranja tehnički koeficijenti i dobiti po jedinici proizvoda odnosno rashodi. Tehnički koeficijenti a_{ij} pokazuju koliko je sati rada i-tog stroja potrebno za obradu jedne jedinice j-tog proizvoda. Uz osnovne varijable x_j koje se još nazivaju i strukturnim varijablama javljaju se još i dva oblika varijabli koje su nam potrebne kako bi mogli linearni problem riješiti pomoću simpleks metode. Ovdje ih se nije posebno definiralo jer će o njima biti riječi u sljedećem poglavlju gdje će se detaljno objasniti zašto se i njih mora unijeti prilikom postavljanja problema.

3.2. Matematičko definiranje problema linearnog programiranja

Centralni matematički problem linearnog programiranja je pronaći rješenje sustava linearnih jednakosti koje maksimiziraju ili minimiziraju datu linearnu funkciju cilja [Ravindran, Phillips, Solberg, 2007, 30]. S matematičkog gledišta razlikuju se tri oblika problema linearnog programiranja [Dobrenić, 1966, 25]:

1) **standardni problem**
2) **kanonski problem**
3) **opći problem**

Uz navede oblike javlja se još i četvrti problem, problem transporta. O njemu se neće pisati u okviru ovog rada, ali on svakako ima bitnu ulogu unutar linearnog programiranja i

kao metoda sama za sebe. Problemi linearnog programiranja zadaju se u standardnom ili općem obliku. Opći oblik je za neke metode potreban svesti na standardni kako bi se određena tehnika rješavanja mogla primijeniti. Za slučaj ovoga rada odnosno simpleks metode oblik kojem će se najviše pažnje posvetiti je kanonski oblik.

3.2.1. Standardni problem i dualnost

Standardni problem linearnog programiranja može se podijeliti u dvije kategorije. Za minimum i za maksimum. Ovisno što se želi optimizacijom postići odnosno želimo li funkciju cilja minimizirati ili pak maksimirati. Ove se dvije kategorije onda razlikuju upravo u tim ograničenjima i ciljevima koji su im postavljeni. Preko sljedeće tablice prikazano je kako problem mora biti postavljen da bi bio u standardnom obliku. Matematički izrazi su već postavljeni u prethodnom poglavlju, ali nisu bili razdijeljeni i prikazani na ovakav način.

Tablica 2. Standardni problemi za max i min [Izvor: Dobrenić, 1975]

Standardni oblik LP za maksimum	**Standardni problem LP za minimum**
Maksimizirati	**Minimizirati**
$\boldsymbol{Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \rightarrow max}$	$\boldsymbol{Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \rightarrow min}$
Uz ograničenja	
$\boldsymbol{a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1}$ $\boldsymbol{a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2}$ $\boldsymbol{\ldots\ldots\ldots\ldots\ldots\ldots}$ $\boldsymbol{a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m}$	$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq b_1$ $a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \geq b_2$ $\ldots\ldots\ldots\ldots\ldots\ldots$ $a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geq \boldsymbol{b_m}$
i uvjet nenegativnosti	
$\boldsymbol{x_1, x_2, \ldots, x_n \geq 0}$	

Prilikom definiranja standardnog oblika problema mora se spomenuti i činjenica da se zadaci linearnog programiranja javljaju u parovima. Tu činjenicu nazivamo princip dualiteta [Lončar, Hunjak, 1978, 235]. Svakom standardnom problemu linearnog programiranja za maksimum korespondira određeni problem za minimum i obratno. Originalni problem koji se još naziva i primal simetričan je dualnom obliku [Dobrenić, 1975, 30]. Unutar ovog rada neće se dublje ulaziti u njegovu upotrebnu i analiziranje, ali ono što svakako treba spomenuti je da je dualitet nešto što je izrazito bitno onome koji provodi analizu – jer mu daje dublji uvid u problem i bolju podlogu za donošenje odluka. Kroz jedan primjer kasnije napomenut će na koji način se interpretira rješenje dobiveno u dualu i što ono znači za dati primjer. Zanimljivo je i vrlo važno za reći da metode linearnog programiranja rješavaju oba problema simultano

pa tako i simpleks metoda dolazi jednako do rješenja kako u primalu tako i u dualu. Nije potrebno rješavati jedno pa drugo, već vrijednosti funkcije cilja za oba oblika bit će jednake i iskazane u završnoj optimalnoj iteraciji simpleks tablice – naravno ukoliko takvo optimalno rješenje postoji.

Dual originalnog problema formira se tako da lijeva strana nejednadžbi ograničenja s pripadajućim vrijednostima i varijablama, koja se primalu čitala horizontalno, postavi u stupce jedan za drugim isto tako kako su ograničenja u originalu išla jedan za drugim u redovima. Vrijednosti ograničenja originala postat će konstante funkcije cilja duala, a konstante funkcije cilja originala postat će vrijednosti ograničenja duala. Dakle, original ima $\boldsymbol{m}$ ograničenja i $\boldsymbol{n}$ varijabli, dok dual ima $\boldsymbol{n}$ ograničenja i $\boldsymbol{m}$ varijabli [Loomba, 1964, 149]. Sukladno s time mijenjaju se i znaci nejednakosti i ciljevi optimiranja. Pa stoga ak je originalni problem bio u standardnom obliku za maksimum njegov dual bit će izražen kao standardni oblik za minimum. Strukturne varijable duala označavaju se s $\boldsymbol{y_i}$. Sada će se sve to objasniti uz kratki primjer iz poglavlja 3.1:

maksimizirati $\quad \mathbf{8}x_1 + \mathbf{3}x_2 \,,$

uz uvjete

$$\mathbf{5}x_1 + \mathbf{3}x_2 \leq \mathbf{30}\,,$$
$$\mathbf{4}x_1 + \mathbf{0}x_2 \leq \mathbf{12}\,,$$
$$x_1, x_2 \geq 0\,.$$

Navedeni original koji je zadan u standardnom obliku prikažimo u dualnom:

minimizirati $\quad \mathbf{30}y_1 + \mathbf{12}y_2 \,,$

uz uvjete

$$\mathbf{5}y_1 + \mathbf{4}x_2 \geq \mathbf{8}\,,$$
$$\mathbf{3}y_1 + \mathbf{0}x_2 \geq \mathbf{3}\,,$$

i uvjete nenegativnosti koji se također moraju postaviti

$$y_1, y_2 \geq 0\,.$$

Rješenje ovog sustava odnosno problema iznosi $y_1 = 1,\ y_2 = \frac{3}{4}$. Funkcija cilja tada poprima vrijednost $Z' = 30 \cdot 1 + 12 \cdot \frac{3}{4} = \mathbf{39}$, što je isti iznos kao i iznos funkcije cilja u originalu. Rješenja varijabli duala koja su se dobila pokazuju koliko neto prihoda donosi jedan radni sat stroja u okviru danog programa [Dobrenić, 1975, 26]. Tako jedan radni sat

prvog stroja donosi 1 n.j., dok jedan radni sat drugog stroja donosi manje i to 0,75 n.j. neto prihoda. Iz toga se može zaključiti da ukoliko se poveća kapacitet prvog stroja za jedan radni sat, maksimalni neto prihod programa će se povećati za 1 n.j. Poveća li se kapacitet drugog stroja isto za jedan radni sat, maksimalni neto prihod povećat će se za 0,75 n.j.

Ovim jednostavnim primjerom pokazano je kako vrijednosti varijabli duala omogućuju i pomažu procjeniteljima i stručnjacima u dobivanju boljeg uvida u to koliko im koji stroj 'vrijedi' i za koji je poželjnije da mu se poveća ili smanji broj radnih sati obrade proizvoda. U stvari vrijednosti varijabli duala pokazuju 'težinu' pojedinih ograničenja koja predstavljaju 'usko grlo' optimalnog programa. Vrijednost varijable duala bit će jednaka nuli ukoliko kapacitet na koji se ona odnosi ne bude u potpunosti iskorišten dobivenim programom. Takav kapacitet odnosno ograničenje ne predstavlja 'usko grlo' proizvodnje.

Kako se uz linearno programiranje kao matematičku metodu vežu mnogobrojni teoremi i definicije tako svakako i uz dual. Unutar ovog rada neće se direktno dokazivati ti teoremi, ali oni su svakako temelj preko kojeg se dolazi do pravila pojedine metode. Na temelju fundamentalnog teorema dualiteta, zaključuje se da postoje četiri moguća slučaja koja se mogu pojaviti kod rješavanja problema linearnog programiranja [Babić, 2010, 78]:

a) Oba problema imaju optimalno rješenje i optimalne vrijednosti funkcija cilja su jednake,
b) Originalni problem nema moguće rješenje, pa dualni nema optimalno,
c) Originalni problem ima moguće, ali nije optimalno rješenje iz čega slijedi da dual nema moguće rješenje,
d) Ne postoji rješenje ni na jednoj strani jer su uvjeti nekonzistentni, tj. sistemi nejednadžbi su u sebi kontradiktorni.

Neki od ovih oblika slučajeva mogućih rješenja spomenut će se kasnije i malo detaljnije razraditi, ali samo u okviru originalnog problema. Za sada je dovoljno ovo napomenuti.

3.2.2. Kanonski problem

Kanonski problem linearnog programiranja razlikuje se od standardnog problema u tome da su sva ograničenja, osim onih kojima se postavlja uvjet nenegativnosti, u obliku jednadžbi [Babić, 2010, 88]. Ovaj oblik kako je već rečeno pogodan je za primjenu različitih metoda rješavanja problema linearnog programiranja. Upravo zato koristit će se i u ovom radu za rješavanje problema simpleks algoritmom. Kanonski problem polazi od toga da se

originalni problem mora svesti na jednakosti. To se postiže uvođenjem novih **dopunskih i/ili artificijelnih varijabli**. Za svaki znak nejednakosti, ali i jednakosti, prikazat će se na koji se način uvode ove promjene.

Da bi se nejednadžbu u obliku „manje od ili jednako“ - ≤ dovelo u oblik jednakosti potrebno joj je lijevoj strani dodati varijablu koja će to i omogućiti. Takva varijabla zove se dopunska varijabla i označavat će se s $\boldsymbol{u}$**.** Dopunskih varijabli bit će onoliko koliko je i nejednadžbi unutar zadanog problema. Indeksi dopunski (ali i artificijelnih) varijabli dodjeljuju se onako kako se pojavljuju u problemu odnosno u ograničenjima – bit će objašnjeno kroz primjere. Ove varijable također ne smiju biti negativne, pa stoga eventualna njihova pozitivna vrijednost ne smije utjecati na vrijednost funkcije cilja [Vučković, 1983, 53]. Unutar ograničenja koja su ovako postavljena, ove varijable pokazuju koliko je po optimalnom programu ostalo neiskorištenog kapaciteta pojedinog stroja ili materijala ili što je već u zadatku zadano. Pošto funkcija cilja mjeri prihod ili rashod npr. neiskorištenih 12 sati rada na nekome stroju ne pridonosi prihodu. Prihod funkcije mora zavisiti samo od vrijednosti varijabli x_j odnosno koliko se proizvoda za dati program proizvodi. Zato se svaka od dopunskih varijabli u funkciju cilja uvodi s koeficijentom 0. Ovakve dopunske varijable nazivaju se još i kao neiskorištene, oslabljenje ili slobodne varijable [Vučković, 1983, 55], ali unutar ovog rada zvat će ih se samo dopunskim. Ukoliko je u nekom mogućem rješenju problema za dopunsku varijablu u ovakvom ograničenju dobiveno 0 znači da takvo ograničenje u potpunosti iskorišteno.

Nadalje, da bi se nejednadžba u obliku „veće od ili jednako“ - ≥ dovela u oblik jednakosti lijevoj je strani pak potrebno dodati varijablu s negativnim predznakom koja će to isto omogućiti. Radi se isto o varijabli $\boldsymbol{u_i}$**.** Te dodatne varijable negativnog predznaka nazivaju se još i varijable viška s obzirom na to da pokazuju koliko je lijeva strana ograničenja veća od desne [Babić, 2010, 90]. Znači da ukoliko je kod nekog mogućeg rješenja problema neka dopunska varijabla veća od 0 ($u_i > 0$) znači da lijeva strana ovakve pripadne ograničavajuće nejednakosti ima veću vrijednost od odgovarajuće desne strane i da je takvo ograničenje u potpunosti iskorišteno.

Uz dopunske varijable negativnog izraza uvijek u paru dolaze i njihove pripadajuće artificijelne varijable $\boldsymbol{w}$ pozitivnog predznaka. Još ih se naziva i umjetne varijable. Njih se uvodi iz razloga što zbog negativnih predznaka dopunskih varijabli ne može dobiti početno nenegativno bazično rješenje. O početnim bazičnim rješenjima će se govoriti nešto kasnije, ali bitno je sada napomenuti kako želimo izbjeći takva rješenja jer su ona nemoguća i

kontradiktorna. Ovim varijablama će se u funkciji cilja pridružiti nespecifično veliki broj $\boldsymbol{M}$. Razlog ovomu je što ovakve varijable služe za lakše rješavanje problema i nikako ne smiju utjecati na funkciju cilja. Iz tog razloga njihove vrijednosti sežu u beskonačnost i ne smiju ni na koji način biti 'kandidati' za moguće rješenje. Kako se u problemu za maksimum ima za cilj dobiti što veća konačna vrijednost tako će se prilikom definiranja funkcije cilja u kanonskom obliku dodati uz varijable w ovaj dovoljno veliki broj M, ali u negativnom izrazu kako bi varijablu uz njega postavili u što inferiorniji odnos spram ostalih varijabli.

Početno bazično rješenje sadrži samo (pozitivne) dopunske i artificijelne varijable odnosno početno rješenje se postavlja tako da mogućih rješenja varijabli mora biti onoliko koliko i ograničenja. Stoga se i u kanonskom problemu ograničenjima zadanim s jednakostima mora uvesti isto artificijelna varijabla kako bi osigurali tom ograničenju ulazak u bazu. Svakako da ta varijabla ne smije utjecati na funkciju cilja i rješenje odnosno ne smije se uopće uzimati kao moguće rješenje, jer takvo ne postoji, pa prema tome i tim varijablama koje su dodane ograničenjima izraženim jednadžbama mora se pridružiti dovoljno veliki broj kako se one ne bi razmatrale kao moguće rješenje. U ovisnosti o cilju funkcije isto se gleda hoće li on ići u minus beskonačno ili plus beskonačno. Također artificijelne varijable isto ulaze u uvjet nenegativnosti kao i dopunske.

Vrlo je vjerojatno da sve rečeno za kanonski oblik nema trenutno prevelikog smisla, ali sve će to 'sjesti na svoje mjesto' preko primjera koji će se razraditi u ostalim poglavljima. Za sada za lakši prikaz ovog oblika poslužit će sljedeća tablica:

Tablica 3: Dopunske i artificijelne varijable

Ograničenje	**Varijable**
$\leq$	$+u$
$\geq$	$-u+w$
$=$	$+w$
Funkcija cilja	
$Z \rightarrow max$	$-M$
$Z \rightarrow min$	$+M$

3.2.3. Opći problem

Ukoliko je problem linearnog programiranja zadan u općem problemu znači da su u tom problemu neovisno je li zadan za maksimum ili minimum ograničenja izražena nejednadžbama $\leq$ i/ili $\leq$ ili pak i jednadžbama $=$ odnosno izražena su sustavom jednadžbi i

nejednadžbi. Takav problem potrebno je svesti na standardni odnosno kanonski problem ukoliko ga se želi riješiti odabranom metodom linearnog programiranja.

3.3. Ekonomsko definiranje problema linearnog programiranja

Kako se već reklo, u realnom slučaju gdje se najviše koriste metode linearnog programiranja, problemi se javljaju u okviru ekonomije. Neki od takvih problema su problemi strukture proizvodnje, problemi strukture sredstava i izvora sredstava, problemi sastava smjesa, problemi zaliha proizvoda itd. Takvi ekonomski problemi, ukoliko ih je moguće, treba svesti na matematički. Ekonomski problem koji ima sljedeće tri kvantitativne komponente može se definirati kao problem linearnog programiranja [Dobrenić, 1966, 31]:

a) cilj i kriterij
b) alternativne metode ili procese za postizanje cilja (kriterija)
c) ograničene resurse i druge uvjete.

Odnosno da bi se ekonomski problem mogao tretirati kao problem linearnog programiranja, mora zadovoljavati sljedeće matematičke uvjete [Dobrenić, 1975, 36]:

1. linearnost funkcije kriterija i sistema jednadžbi odnosno nejednadžbi
2. diskretnost procesa odnosno aktivnosti
3. zbrojivost procesa u utrošku resursa i u funkciji cilja
4. proizvoljna djeljivost faktora
5. ograničeni broj procesa odnosno aktivnosti i ograničenja

Ne bi se trebalo nešto puno ovdje reći jer navedene definicije odnosno uvjeti su već u većoj ili manjoj mjeri rečeni i prilikom matematičkog definiranja. Ono na čemu će se ovaj rad bazirati je matematički dio linearnog programiranja i kako pomoću njega doći do rješenja. Duboka i detaljna tumačenja tih rješenja u ekonomskom smislu neće se obrađivati.

3.4. Rješavanje problema linearnog programiranja

Problemi linearnog programiranja rješavaju se metodama linearne algebre. Matematička sredstva koja se moraju upotrijebiti pri rješavanja takvog problema ovisna su o obimnosti problema, pogotovo o broju raspoloživih varijabli i broju uvjeta u formi jednadžbi odnosno nejednadžbi [Martić, Vandal, 1968, 8]. Po opsegu veoma mali problemi mogu se rješavati grafičkim metodama ili nekim pak elementarnim numeričkim aritmetičkim metodama. Metoda na koju se ovaj rad oslanja je simpleks metoda. Metoda pomoću koje se mogu rješavati kompleksni i zahtjevni problemi koji su toliko veliki da ih se jednostavno

mora riješiti i uz pomoć računala jer mogu sadržavati i po više stotina varijabli. Svim linearnim problemima postupak rješavanja je isti i on se sastoji od sljedećih koraka [Dobrenić, 1975, 39]:

1. definiranje problema
2. prikupljanje i sređivanje informacija i podataka
3. postavljanje modela
4. rješavanje problema
5. testiranje dobivenog programa
6. interpretacija dobivenog konačnog – optimalnog programa i eventualnih suboptimalnih programa.

Ovi koraci čine se jednostavnima, ali u realnim primjerima unutar stvarnih organizacija prvi koraci – 1, 2 i 3 - su ključni. Unutar njih se pokušavaju prikupiti svi relevantni podaci te otkloniti eventualni problemi ili nesmislenosti koji bi mogli dovesti do krivih rješenja. Koraci 4 i 5 provodit će se u ovome radu. Nešto malo kroz jedan kratki primjer i korak 3 preko kojeg će se iz ekonomskog modela postaviti potrebiti matematički model. Formulacija takvog modela provodi se kroz sljedeća tri koraka [Ravindran, Phillips, Solberg, 2007, 14]:

korak 1. Identificirati nepoznate varijable koje će se rješavati i prikazati ih pomoću algebarskih simbola

korak 2. Identificirati sva ograničenja i uvjete u problemu i izraziti ih kao linearne jednadžbe ili nejednadžbe koje su linearne funkcije nepoznatih varijabli

korak 3. Identificirati cilj ili kriterij i predstaviti ga kao linearnu funkciju varijabli odlučivanja, koja treba maksimizirati ili minimizirati

Svakako treba naglasiti kako odabrana simpleks metoda sama provodi testiranje dobivenih rješenja kroz izvođenja njenog algoritma. Korak testiranja više je potreban kod ručnog računanja nego putem računalnih programa, ali svakako je korak kojeg se ne smije nikako preskočiti i koji vodi točnoj interpretaciji dobivenog programa.

Sljedeći primjer[2] koji će se riješiti grafičkom metodom, iz razloga jer opisuje dvije nepoznate varijable i stoga ga je moguće prikazati u dvodimenzionalnom prostoru, pojasnit će što se točno traži i na koji način tijekom rješavanja linearnog programiranja. Slučaj koji je

[2] Zadatak preuzet iz materijala dostupnih na e-kolegiju 'Operacijska istraživanja 1' – riješen samostalno

uzet ima jedno i optimalno rješenje, koje naravno nije uvijek prisutno u većim i kompleksnijim problema. O posebnim slučajevima nešto kasnije.

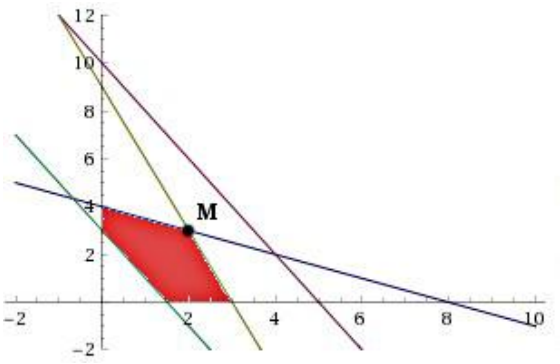
Primjer se sastoji od sljedeće funkcije cilja i njenih ograničenja i uvjeta odnosno model za ovaj primjer glasi:

$$\boldsymbol{Z = 3x_1 + 2x_2 \rightarrow max}$$

$$\begin{aligned} \boldsymbol{P_1} \quad & 3x_1 + 6x_2 \leq 24 \\ \boldsymbol{P_2} \quad & 8x_1 + 4x_2 \leq 40 \\ \boldsymbol{P_3} \quad & 9x_1 + 3x_2 \leq 27 \\ \boldsymbol{P_4} \quad & 2x_1 + 1x_2 \geq 3 \end{aligned}$$

$\boldsymbol{x_1, x_2 \geq 0}$ → uvjet nenegativnosti**!**

Svakom ograničenju dodijelile su se oznake za pravce kako bi ih se lakše raspoznalo i nacrtalo unutar grafa odnosno sva ograničenja unutar grafičkog rješenja crtaju se kao pravci sa svojim područjem rješenje – koje se nalazi s jedne od strana istog. Ukoliko je ograničenje zadano jednadžbom područje rješenja za taj pravac svakako je on sam. Rješenje programa se traži unutar prvog kvadranta. Odnosno ukoliko rješenje nije unutar prvog kvadranta ono zasigurno ne može biti optimalno rješenje, jer znači da barem jedna točka rješenje nije unutar uvjeta nenegativnosti. Grafičko rješenje[3] ovog programa izgleda ovako:



Slika 1. Grafičko rješenje problema

Zadatak linearnog programiranja je da od beskonačno mnogo točaka izabere jednu (ili možda više njih) za koju zadana funkcija cilja ima maksimalnu (ili minimalnu) vrijednost [Babić, 80]. I stoga na području mogućih rješenja treba naći onu točku koja unutar tog područja rješenja zadovoljava funkciju cilja, odnosno treba pronaći ekstremnu točku područja bilo minimalnu ili maksimalnu. U ovom primjeru područje rješenja je pobojano crvenom bojom. Unutar tog područja treba pronaći ekstremnu točku programa koja će dati maksimalni mogući rezultat. Ona se u grafičkom načinu rješavanja linearnog problema nalazi tako da se

[3] Izrađen samostalno u: https://www.wolframalpha.com

pravac funkcije cilja povlači ka rubovima područja rješenja dok se dostigne do one u ovom slučaju maksimalne. U ovom primjeru ta maksimalna točka nalazi se na sjecištu pravca P_1 i P_3 i označila se kao točka M. Rješavanje sjecišta dvaju pravaca daje koordinate te točke $M(2,3)$, odnosno rješenja za x_1 i x_2, a ona glase:

$$\boldsymbol{x_1 = 2}\,,$$

$$\boldsymbol{x_2 = 3}\,.$$

Dok je vrijednost funkcije cilja s ovakvim programom jednaka:

$$\boldsymbol{Z} = 3 \cdot 2 + 2 \cdot 3 = \mathbf{12}\,.$$

Ono što se ovim primjerom htjelo pokazati je način na koji se traži ta ekstremna točka. Simpleks metoda na isti način u svim mogućim rješenjima pokušava doći do one točke n-dimenzionalnog prostora koja će zadovoljiti funkciju cilja odnosno polučiti maksimalnoj ili minimalnoj vrijednosti iste. Kako zadani problem sadrži samo dvije nepoznanice njega je bilo moguće riješiti grafičkom metodom dok kod problema s više varijabli to je malo kompliciranije. Zato se u takvim slučajevima primjenjuje simpleks metoda.

4. Simpleks algoritam

Simpleks algoritam sastavni je dio simpleks metode koja je pak temelj linearnog programiranja. Sama metoda razvila se 1947. godine od strane G. Dantziga odnosno 1951. kada je i objavio osnovni rad o njoj [Martić, 1979, 103]. Iako za razne specijalne slučajeve postoje brže i bolje metode, simpleks metoda je i dan danas 'kraljica' linearne optimizacije i po jednostavnosti i po broju korisnika [Čaklović, 2010, 27]. Ovaj rad obrađuje striktno algebarski dio simpleks metode i to onaj koji se rješava putem tabelarnog/tabličnog prikaza, ali bitno je za naglasiti kako ona ima veliko uporište u geometrijskoj matematici i da su joj preko nje postavljeni temelji. Odnosno, simpleks metoda rješava problem unutar okvira linearne algebre, ali ima lijepu geometrijsku interpretaciju [Čaklović, 2010, 27].

Simpleks metoda linearnog programiranja zahtjeva da se prvo odredi neko polazno bazično rješenje odnosno kompletira odgovarajuća polazna simpleks tabela [Vučković, 109]. Zatim se odgovarajućim testovima takvo polazno bazično rješenje unapređuje sve do optimalnog rješenja. To unapređenje odvija se postupno – iterativno izmjenom tekuće bazične kombinacije u novu bazičnu kombinaciju koje se međusobno razlikuju samo u jednoj bazičnoj varijabli dok im je m-1 ostalih varijabli istih. Postupak takvog rješavanja obavlja se preko odgovarajućeg niza simpleks tabela.

Prilikom primjene simpleks metode odnosno algoritma bitno je razlikovati simpleks metodu koja polazi s linearnim programom u standardnom obliku i simpleks algoritam koji polazi s kanonskim oblikom, a sastoji se od niza stožernih operacija i čini glavni dio subrutine simpleks metode [Dantzig, 1963, 94]. Stoga da bi se primijenio simpleks algoritam mora se prvo unutar metode standardni odnosno opći oblik pretvoriti u kanonski oblik. Kako je već rečeno kanonski oblik dobiva se tako da se sve nejednadžbe unutar originalnog postavljenog problema, bio on zadan u standardnom ili općem obliku, pretvore u sustav jednadžbi tako što će mu se dodati odgovarajuće dopunske varijable **(u)** i/ili artificijelne varijable **(w).** I tek tada kada se dobio kanonski oblik može se krenuti s provođenjem simpleks algoritma, koji je se u ovisnosti o optimizaciji (max ili min) razlikuje u par koraka. Stoga će se prvo krenuti od rješavanja standardnog problema linearnog programiranja simpleks algoritmom za maksimum. Zatim će se objasniti postupak rješavanja standardnog problema linearnog programiranja za minimum, te će se na kraju objasniti i problemi postavljeni općim oblikom. Uz svaki od navedenih primjera postepeno će se uz rješavanje objašnjavati odnosno razvijati i nadograđivati i sam algoritam.

4.1. Standardni problem linearnog programiranja za maksimum

Kako je već navedeno standardni problem linearnog programiranja za maksimum izgleda tako da su mu sva ograničenja zadana gornjom granicom odnosno znakom nejednakosti „manje od ili jednako“. Za prikazivanje rješavanja ovakvog oblika problema uzet će se sljedeći primjer[4]:

$$\boldsymbol{Z = 75x_1 + 85x_2 + 55x_3 + 95x_4 \rightarrow max}$$

$$0x_1 + 1x_2 + 3x_3 + 2x_4 \leq 750$$
$$4x_1 + 0x_2 + 0x_3 + 1x_4 \leq 350$$
$$1x_1 + 1x_2 + 4x_3 + 2x_4 \leq 450$$
$$x_1, x_2, x_3, x_4 \geq 0$$

Naravno, prvi korak koji se mora provesti unutar simpleks metode je problem koji je zadan prevesti u kanonski oblik. Unutar njega na već spomenuti način unose se dopunske varijable koje će se zapisivati po dijagonali kako se pojavljuju iz razloga jer će se na taj način kasnije i unositi u tablicu kao jedinična matrica. Kanonski problem stoga izgleda ovako:

$$\boldsymbol{Z = 75x_1 + 85x_2 + 55x_3 + 95x_4 + 0(u_1 + u_2 + u_3) \rightarrow max}$$

$$\begin{aligned} 0x_1 + 1x_2 + 3x_3 + 2x_4 + \boldsymbol{u_1} &= 750 \\ 4x_1 + 0x_2 + 0x_3 + 1x_4 + \boldsymbol{u_2} &= 350 \\ 1x_1 + 1x_2 + 4x_3 + 2x_4 + \boldsymbol{u_3} &= 450 \end{aligned}$$
$$x_1, x_2, x_3, x_4, \boldsymbol{u_1, u_2, u_3} \geq 0$$

Iz priloženog vidimo kako su dopunske varijable ušle u funkciju cilja s koeficijentom nula iz razloga koji je već naveden, a taj je da one ne smiju imati utjecaja na njenu vrijednost – ne pridonose nikakav prihod. Koeficijenti koji se tim varijablama dodaju unutar ograničenja pri formiranju početnog rješenja je jedan[5] i time formiranju jediničnu matricu. Na temelju ovih podataka odnosno kanonskog oblika formira se prva početna simpleks tablica. Tu početnu tablicu gledamo kao prvu fazu simpleks algoritma, a ona za cilj ima pronalaženje bilo kakvog **bazičnog mogućeg rješenja.**

Bazično rješenje unutar tablice služi nam kao sredstvo pomoću kojeg dolazimo odnosno tražimo optimalno moguće rješenje. Na koji način se to radi pokazat će se preko primjera. Bazično rješenje sadrži točno m pozitivnih vrijednosti, dakle onoliko koliko postoji nezavisnih jednadžbi ograničenja [Dobrinić, 1975, 51]. Najbolje moguće bazično rješenje

[4] Zadatak preuzet iz materijala dostupnih na e-kolegiju 'Operacijska istraživanja 1' – riješen samostalno
[5] za to ograničenje u kojem se prvi put pojavljuje, unutar ostalih ograničenja ono iznosi isto nula

zove se **optimalno rješenje problema** i to je rješenje kod koga funkcija cilja dostiže željenu ekstremnu vrijednost (maksimalnu ili minimalnu – zavisno od cilja) koja mora biti konačna [Vučković, 1983, 82]. Svaka od simpleks tablica u nizu odgovara bazičnom rješenju problema koje je bliže njegovom optimalnom rješenju [Vučković, 1983, 105]. Naravno da rješenje ne morao uvijek ispasti optimalno i jedinstveno, ali za sada je ovo dovoljno reći za bazično rješenje. Unutar poglavlja posebnih svojstava simpleks algoritma više će se pozabaviti različitim slučajevima vezanim za bazična rješenja.

Svaki kanonski oblik zapravo u sebi sadrži to prvo bazično rješenje koje je potrebno za postavljanje prve početne simpleks tablice. Simpleks tablica formira se na sljedeći način:

Tablica 4[6]. Početna tablica simpleks metode za problem maksimuma

	$c_j \rightarrow$		c_1 $c_2 \cdots \boldsymbol{c_s} \cdots c_n$	0 $0 \cdots 0$ $\cdots 0$		
$\boldsymbol{c_s}$ ↓	**Bazično rješenje**		**Strukturne varijable**	**Dopunske varijable**		
	Varijabla	**Količina**	x_1 $x_2 \cdots \boldsymbol{x_s} \cdots x_n$	u_1 $u_2 \cdots u_r \cdots u_m$	**K**	**R**
0	u_1	a_{10}	a_{11} $a_{12} \cdots \boldsymbol{a_{1s}} \cdots a_{1n}$	1 $0 \cdots 0$ $\cdots 0$		
0	u_2	a_{20}	a_{21} $a_{22} \cdots \boldsymbol{a_{2s}} \cdots a_{2n}$	0 $1 \cdots 0$ $\cdots 0$		
⋮	⋮	⋮	⋮	⋮ ⋮ ⋮ ⋮		
← **0**	$\boldsymbol{u_r}$	$\boldsymbol{a_{r0}}$	$\boldsymbol{a_{r1}}$ $\boldsymbol{a_{r2}} \cdots \boldsymbol{a_{rs}} \cdots \boldsymbol{a_{rn}}$	**0** **0** $\cdots$ **1** $\cdots$ **0**		
⋮	⋮	⋮	⋮	⋮ ⋮ ⋮ ⋮		
0	u_m	a_{m0}	a_{m1} $a_{m2} \cdots \boldsymbol{a_{ms}} \cdots a_{mn}$	0 $0 \cdots 0$ $\cdots 0$		
	$\boldsymbol{Z_j - c_j}$	0	$-c_1$ $-c_2 \cdots -\boldsymbol{c_s} \cdots -c_n$	0 $0 \cdots 0$ $\cdots 0$		

Tablica unutar zaglavlja u prvom redu sadrži koeficijente varijabli $\boldsymbol{c_j}$ iz funkcije cilja. Prvi stupac $\boldsymbol{c_s}$ sadrži koeficijente iz funkcije cilja onih varijabli koje se nalaze u bazičnom rješenju. Kako se prvo početno moguće bazično rješenje formira iz kanonskog oblika vrijednosti ovih koeficijenata za početnu tablicu će naravno iznositi 0 pošto se to prvo bazično rješenje sastoji od dopunskih varijabli. Za popuniti do kraja to prvo bazično rješenje moraju se unijeti te dopunske varijable u drugi stupac tablice koji inače pokazuje koje su trenutno varijable uzete za rješenje problema. Treći stupac sadrži njihove trenutne vrijednosti.

[6] Prema: Dobrenić, 1975, 53.

Kako su u početnom bazičnom rješenju kreće od toga da su strukturne varijable $\boldsymbol{x_i}$ jednake 0 tako će vrijednosti trećeg stupca biti vrijednosti postavljenih ograničenja problema. Kako bi se kasnije prikazao korak transformacija, za sada se te vrijednosti neće označavati kao prije s b_i već će im se za početni indeks dodijeliti $j = 0$ uz oznaku a odnosno $b_1 \equiv \boldsymbol{a_{10}}, b_2 \equiv \boldsymbol{a_{20}}$ i tako redom [Dobrenić, 1975, 55].

Središnji dio tablice sadrži matricu input-output koeficijenata $\boldsymbol{a_{ij}}$ strukturnih varijabli, desni dio tablice sadrži već spomenutu jediničnu matricu dopunskih varijabli.

Zadnji red tablice $c'_j = Z_j - c_j$ predstavlja red kriterija optimalnosti. Odnosno taj redak tj. njeni koeficijenti koji se još nazivaju i relativni troškovi ukazuju za koliko će Z biti uvećan (ili umanjen) za svako jedinično povećanje varijable koja se nalazi iznad tog koeficijenta [Ravindran, Phillips, Solberg, 2007, 34]. Iz tog razloga varijabla koja se trenutno nalazi u bazičnom rješenju uvijek će u tom retku poprimati vrijednost 0. No nužno ne mora to biti jedina varijabla koja ima tu vrijednost. O tome što to znači reći će se nešto kasnije. Predzadnji stupac - kontrolni stupac – nije nužno provoditi jer ne predstavlja nužan uvjet za donošenje rješenja, ali korisno ga je koristiti prilikom ručnog računanja jer kako mu i sam naziv govori koristi se za kontrolu izračunatog. On sadržava zbroj svih vrijednosti za red na koji se odnosi. U početnoj tablici na taj način se i računa dok u ostalim tablicama na jedan drugačiji način. Detaljnije o tome nešto kasnije. O zadnjem stupcu R također nešto više u nastavku.

Na temelju svega prethodno rečenog popunjava se početna tablica i ona sada izgleda ovako:

Tablica 5. Početna simpleks tablica za primjer 1.

$c_j \rightarrow$			**75**	**85**	**55**	**95**	**0**	**0**	**0**		
c_s ↓	**Bazično rješenje**		**Strukturne varijable**				**Dopunske varijable**				
	Var	**Kol**	x_1	x_2	x_3	x_4	u_1	u_2	u_3	**K**	**R**
0	u_1	**750**	0	1	3	2	1	0	0	757	
0	u_2	**350**	4	0	0	1	0	1	0	356	
0	u_3	**450**	1	1	4	2	0	0	1	459	
	$z_j - c_j$	**0**	**-75**	**-85**	**-55**	**-95**	**0**	**0**	**0**	-310	

Sada kada se popunila početnu tablicu, završena je prva faza simpleks algoritma. Kako ovo bazično rješenje sadrži samo dopunske varijable vrlo je jasno kako ono nije i optimalno rješenje stoga započinje druga faza simpleks algoritma. Faza unutar koje se provodi optimizacija i pomoću koje se dolazi do optimalnog maksimalnog ili minimalnog rješenja problema, ukoliko takvo i postoji. Druga faza sastoji se od konačnog broja iteracija za poboljšanje početnog bazičnog programa [Dobrenić, 1975, 55]., Svaka iteracija sastoji se od tri sljedeća koraka:

1) **određivanje vodećeg stupca** – unutar ovog koraka određuje se koja će to varijabla ući u bazu kao moguće rješenje problema. Kod ovog koraka za problem maksimuma cilj je izabrati one varijable da uđu u bazu kod kojih će se maksimalno povećati vrijednost funkcije cilja. Kod minimuma naravno izabiru se one koje će smanjivati vrijednost funkcije.

Za odabir tog stupca služi nam zadnji red tablice, red relativnog profita, koji je iskazan obrnutim predznakom odnosno vrijednost koja je iskazana s minus predznakom predstavlja za koliko će se povećati Z za svaku jediničnu promjenu varijable na koju se taj iznosi odnosi. Pozitivna vrijednost označava za koliko će se funkcija cilja smanjiti. Kako problem maksimuma želimo poboljšavati u smjeru povećanja ukupne vrijednosti Z, vodeći stupac za taj problem bit će onaj koji u tom retku poprima najveću negativnu vrijednost odnosno $\boldsymbol{max}\{|\boldsymbol{Z_j - c_j}|\}$ gdje je $\boldsymbol{Z_j - c_j < 0}$. Za navedeni primjer tu najveću negativnu vrijednost poprima četvrta varijabla x_4 te stupac unutar kojeg se ona nalazi označavamo kao vodeći stupac.

2) **određivanje vodećeg reda** – unutar ovog koraka određuje se koja će to varijabla izaći iz baze, a na njeno mjesto ući nova odabrana nebazična varijabla. Ovo odabiranje vodećeg reda zajedno s odabiranjem vodećeg stupca naziva se pivotiranje [Barković, 61]. Stoga element koji se nađe na raskrižju tog vodećeg stupca i retka nazivamo pivot elementom.

Sada stupac R dolazi u razmatranje. Vodeći red bit će onaj koji poprima najmanji kvocijent prvog stupca i odabranog vodećeg stupca. I te razlike zapisuju se unutar stupca R radi lakšeg snalaženja. Odabire upravo onaj najmanji omjer jer varijabla iz baze unutar reda koja ima najmanju tu razliku prva će poprimiti vrijednost 0 ukoliko poraste nebazična varijabla koja se dovodi u bazu [Barković, 21]. Znači da iz baze izlazi ona varijabla na koju najviše odnosno najprije utječe povećanje varijable koja ulazi u bazu. Ukoliko bi se odabrala

neka druga varijabla dogodilo bi se da u bazičnom rješenju imamo negativne vrijednosti, a to se svakako želi izbjeći. Ovo pravilo nazivamo još i pravilom najmanjeg omjera (eng. minimum ratio rule[7]):

$$\Theta = min\left\{\frac{a_{i0}}{a_{is}}\right\}, \qquad za\ sve\ a_{is} > 0\,,$$

gdje indeks ***s*** predstavlja indeks vodećeg stupca, a uvjet govori da se prilikom računanja ovog omjera ne smije dijeliti s 0 ili negativnom vrijednosti. U odabranom primjeru u početnoj tablici vodeći red postaje red unutar kojeg se nalazi bazična varijabla u_3. Jer njegov omjer je bio najmanji u iznosu od 225.

Tablica 6. Određivanje vodećeg stupca i reda za primjer 1.

c_S ↓	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1	**750**	0	1	3	2	1	0	0	757	375
0	u_2	**350**	4	0	0	1	0	1	0	356	350
0	u_3	**450**	1	1	4	**2**	0	0	1	459	**225**
	$z_j - c_j$	**0**	-75	-85	-55	-95	0	0	0	-310	

Nakon što se odredio vodeći stupac i vodeći red odnosno koja nebazična varijabla ulazi u bazično rješenje i koja bazična varijabla izlazi iz njega, slijedi prva iteracija simpleks algoritma odnosno druga tablica. Prvo što se radi je da se na mjesto bazične varijable koja je izašla iz baze stavi nova varijabla koja je ušla u bazu i to tako da vodeći stupac postaje jedinični stupac tako što stožerni element odnosno pivot poprima vrijednost 1 dok svi ostali postaju 0. Nakon toga vrijednosti elemenata iz vodećeg reda dijele se s vrijednošću pivota i izračunate vrijednosti zapišu se u novu tablicu. Te se vrijednosti ne smiju zaokruživati nego se zapisuju u obliku razlomka ukoliko je potrebno. Svaka zaokružena vrijednost polučit će krive zaključke i rješenja, a to svakako nije za cilj. Za odabrani slučaj to sada izgleda ovako:

[7] Ravindran, Phillips, Solberg, 2007, 38-39.

Tablica 7. a) Prva iteracija za primjer 1.

$c_j \rightarrow$			75	85	55	95	0	0	0		
c_s ↓	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1					0					
0	u_2					0					
95	x_4	225	1/2	1/2	2	1	0	0	1/2	459/2	
	$z_j - c_j$	0				0					

Prilikom popunjavanja ostatka tablice radi kraćenja vremena mogu se koristiti sljedeće olakšice računanja. Postoji li u vodećem stupcu prijašnje tablice na nekome mjestu vrijednost 0. Ukoliko da, vrijednosti reda u kojem se našla ta nula, prepisuju se. Isto tako vrijedi i za red. Ukoliko u vodećem redu postoji vrijednost nula, vrijednosti stupca gdje se nalazila ta vrijednost nula, prepisuju se:

Tablica 8. b) Prva iteracija za primjer 1.

c_S ↓	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1					0	**1**	**0**			
0	u_2					0	**0**	**1**			
95	x_4	225	1/2	1/2	2	1	0	0	1/2	459/2	
	$z_j - c_j$	0				0	**0**	**0**			

Pošto u vodećem redu nije bilo vrijednosti 0, nije se ništa moglo prepisati, dok se za prvi i drugi stupac dopunskih varijabli to moglo učiniti. Sljedeći korak popunjavanja ostatka tablice treći je korak simpleks algoritma [Dobrenić, 1975, 55]:

3) transformacija koeficijenata tabele iz $\boldsymbol{a_{ij}}$ u $\boldsymbol{a'_{ij}}$

a. transformacija koeficijenata vodećeg reda

b. transformacija koeficijenata ostalih redova.

Korak 3.a se već izveo na opisan način. Korak 3.b izvodi se na sljedeći način za $i \neq r$, pošto su elementi s indeksom r bili elementi vodećeg reda:

$$a'_{ij} = a_{ij} - a'_{rj} \cdot a_{is}$$

Stoga, nova vrijednost u novoj iteraciji računa se tako da se od stare vrijednosti tog elementa oduzme umnožak nove vrijednosti tog stupca, ali elementa koji je ujedno bio i u vodećem redu, i stare vrijednosti iz vodećeg stupca za taj redak. Po ovom obrascu transformiraju se i vrijednosti $Z_j - c_j$ reda i kontrolnog stupca. Tako za prvu iteraciju po stupcima ove vrijednosti iznose:

Tablica 9. a) prva transformacija za primjer 1

kol:				x_1				x_2			
750	−225	· 2	= **300**	0	−1/2	· 2	= **−1**	1	−1/2	· 2	= **0**
350	−225	· 1	= **125**	4	−1/2	· 1	= **7/2**	0	−1/2	· 1	= **−1/2**
0	−225	· (−95)	= **21375**	−75	−1/2	· (−95)	= **−55/2**	−85	−1/2	· (−95)	= **−75/2**

Tablica 10. b) prva transformacija za primjer 1.

x_3				u_3				K			
3	−2	· 2	= **−1**	0	−1/2	· 2	= **−1**	757	−459/2	· 2	= **298**
0	−2	· 1	= **−2**	0	−1/2	· 1	= **−1/2**	356	−459/2	· 1	= **253/2**
−55	−2	· (−95)	= **135**	0	−1/2	· (−95)	= **95/2**	−310	−459/2	· (−95)	= **42985/2**

Odnosno dobiva se konačno prva iteracija:

Tablica 11. Potpuna prva iteracija za primjer 1.

$c_j \rightarrow$			**75**	**85**	**55**	**95**	**0**	**0**	**0**		
c_s ↓	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1	**300**	**-1**	**0**	**-1**	0	1	0	**-1**	**298**	
0	u_2	**125**	**7/2**	**-1/2**	**-2**	0	0	1	**-1/2**	**253/2**	
95	x_4	225	1/2	1/2	2	1	0	0	1/2	459/2	
	$z_j - c_j$	**21375**	**-55/2**	**-75/2**	**135**	0	0	0	**95/2**	**42985/2**	

Kontrolni stupac sada će poslužiti kao pomagalo u kontroli dobivenih rješenja. Odnosno vrijednost koja je izračunata preko prijašnjeg koraka mora biti ista onoj vrijednosti

koja se dobije zbroje li se sve vrijednosti retka. Ukoliko se ispostavi da se te vrijednosti ne podudaraju vrlo vjerojatno se negdje pogriješilo rijekom računanja.

Nakon što se dobila prva iteracija preispituju se vrijednosti u $\boldsymbol{Z_j - c_j}$ retku. Za slučaj kada je problem iskazan u obliku standardnog problema za maksimum želi se postići da ovaj redak nema negativne vrijednosti, odnosno da su sve vrijednosti $\boldsymbol{Z_j - c_j \geq 0}$. Nadoveže li se to na već rečeno za relativne troškove jasno je zašto ovo pravilo ovako glasi. Ukoliko u problemu za maksimum ne postoji više niti jedna varijabla s negativnom vrijednošću ovog troška znači da se više ne može unaprijediti program jer izabere li se koja druga varijablu kao bazična funkcija cilja neće više rasti već suprotno. Korisno je ovdje za spomenuti kako simpleks algoritam eliminira sva moguća rješenja koja su lošija od trenutnog postavljenog i time efikasnije dolazi do rješenja nego da ispituje rješenje po rješenje je li ono optimalno. A tih rješenja može biti jako puno. Broj mogućih bazičnih rješenja računa se kao [Ravindran, Phillips, Solberg 32]:

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!}\,,$$

odnosno konkretno za ovaj primjer

$$\binom{4}{3} = \frac{4!}{3!\,1!} = 4\,.$$

Ovaj broj ne znači da će i toliko tablica i koraka prilikom računanja biti već koliko je mogućih kombinacija varijabli i njihovih vrijednosti da zajedno zadovoljavaju ograničenja i uvjete.

Kako prema uvjetu za maksimum $\boldsymbol{Z_j - c_j}$ još uvijek postoje negativne vrijednosti, simpleks algoritam provodi se dalje. Ponovo se traži vodeći stupac, koji je sada x_2, a vodeći redak opet je ispao treći pa iz baze izlazi varijabla x_4. Pivot element je $1/2$.

Tablica 12. Vodeći stupac i vodeći red za drugu iteraciju za primjer 1.

c_s ↓	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1	300	-1	0	-1	0	1	0	-1	298	/
0	u_2	125	7/2	-1/2	-2	0	0	1	-1/2	253/2	/
95	x_4	225	1/2	**1/2**	2	1	0	0	1/2	459/2	**450**
	$z_j - c_j$	21375	-55/2	-75/2	135	0	0	0	95/2	42985/2	/

Pošto se u vodećem redu javlja nula prvi red se može prepisati u novoj iteraciji. Također vrijedi i za stupce ispod dopunskih varijabli koje su i unutar bazičnog rješenja. Provedu li se dalje transformacije na već pokazan način dobiva se sljedeće rješenje odnosno potpuna druga iteracija:

Tablica 13. Druga iteracija za primjer 1.

$c_j \rightarrow$			75	85	55	95	0	0	0		
c_s ↓	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1	300	-1	0	-1	0	1	0	-1	298	
0	u_2	350	4	0	0	1	0	1	0	356	
85	x_4	450	1	1	4	2	0	0	1	459	
	$z_j - c_j$	38250	10	0	285	75	0	0	85	38705	

Prikazano rješenje druge iteracije ujedno i je optimalno bazično rješenje jer više ne postoje negativne vrijednosti u zadnjem retku. I time se dobilo konačno rješenje problema odnosno program koji glasi:

$$x_3 = \mathbf{450}$$

$$u_1 = \mathbf{300}$$

$$u_2 = \mathbf{350}$$

$$Z = \mathbf{38250}$$

Navedeni primjer prikazuje kako nije uvijek nužno odabrati varijablu koja ima najveću negativnu vrijednost u zadnjem redu već treba i racionalno razmišljati i gledati na koji način će se vrijednost funkcije cilja mijenjati ovisno o varijabli koja se izabere za ulazak u bazično rješenje. Tom racionalnijem razmišljanju može poslužiti egzaktni kriterij za izbor varijable koja ulazi u novu bazu [Babić, 2010, 128]. Ovdje se nije ispitivala svaka varijabla koja može ući u bazu već je ušla ona koja po običnom kriteriju ima $max|Z_j - c_j|$. Da smo u prvom koraku odmah odabrali varijablu x_2 optimalno rješenje dobili bi već iz prve iteracije. No ovo je dovoljno malen primjer i zadatak da se jedna iteracija više može zanemariti. O egzaktnom kriteriju bit će riječi u poglavlju posebnih slučajeva simpleks algoritma.

Iz simpleks tablice kako je već naznačeno moguće je pročitati i rješenje dualnog problema ukoliko takvo i postoji. Pošto optimalni program za ovaj problem originala postoji,

rješenje duala isto je optimalno i jednake je vrijednosti od 38250 n.j. Vrijednosti varijabli duala čitaju se ispod dopunskih varijabli u zadnjem redu i prema dobivenom rješenju postoji samo jedna vrijednost, a to je $y_3 = 85$. Kako je već bilo i napomenuto, ova vrijednost duala označava za koliko će se ukupni prihod povećati, povećali se kapacitet stroja 3 (npr. ako smo ograničenja označili kao strojeve) za jednu jedinicu odnosno jedan sat. To znači da jedan sat rada na trećem stroju donosi 85 n.j. prihoda. Pošto je rješenje problema takvo da program ukazuje kako je isplatljivo proizvoditi samo proizvod x_2 čiji prihod po jedinici iznosi 85 n.j. logično je onda zašto je ispala ovakva vrijednost duala. Pošto su dopunske varijable u_1 i u_2 ostale pozitivne one ukazuju na to da je ostalo nešto neiskorištenog kapaciteta. Za prvo ograničenje odnosno stroj ono iznosi 300 sati, dok za drugi 350, odnosno jedini proizvod koji se treba proizvoditi po ovom programu uopće se ne dorađuje na drugome stroju jer je njegov kapacitet ostao u potpunosti neiskorišten. Iz toga se može zaključiti da je stroj broj dva višak, no ta dublja analiziranja ostavit će se na nekome kome će ona više značiti. Za interpretaciju ovog rješenja problema dovoljno je navedeno.

4.2. Standardni problem linearnog programiranja za minimum

Kako se prilikom rješavanja standardnog oblika za minimum i općeg problema za minimum i maksimum uvode nove artificijelne varijable, rješavanje putem simpleks tablice odvija se nešto drugačije nego rješavanje standardnog problema za maksimum. Pošto te artificijelne ne smiju nikako ući u rješenje (u suprotnom je ono nemoguće), postoje tri načina otklanjanja ovog problema odnosno tri metode[8]:

1) metoda polaganja na maksimum, tj.dual
2) metoda veliko M
3) 'dvofazna' metoda

Prva metoda primjenjuje se kada je problem zadan u standardnom obliku za minimum. Pošto je njegov dual standardni problem za maksimum jasno je kako se u takvom problemu kada se prevede u kanonski oblik neće javiti artificijelne varijable i da će ga se moći riješiti na način koji je opisan u prethodnom potpoglavlju. Ukoliko postoji optimalno rješenje, to rješenje duala riješit će i problem originala.

Druga metoda naziv je dobila po velikom koeficijentu koju se stavlja uz artificijelne varijable. Ta metoda koristi pravilo unutarnjeg proizvoda za dobivanje vrijednosti relativnih troškova tablice. Neće se detaljno ova metoda objašnjavati jer prilikom rješavanja ovakvog

[8] Izvori: Dobrenić, 1975, 74 i Kalpić, Molnar, 1996, 18.

problema koristit će se treća navedena metoda. Dovoljno je reći da obje metode 2) i 3) imaju za cilj što prije otkloniti i izbaciti artificijelne varijable iz bazičnog rješenja jer takvo rješenje je daleko od optimalnog i nikako nije prihvatljivo kao jedno od mogućih rješenja. Također obje metode imaju i jednak broj iteracija.

Treća metoda tkz. 'dvofazna metoda' poslužit će za ovaj primjer. Naziv je dobila po svoje dvije faze koje se provode kako bi se dobio program problema [Dobrenić, 1975, 77]:

1. faza: izračunavanje prvog upotrebljivog programa
2. faza: izračunavanje optimalnog programa

Prva faza se sastoji u izračunavanju prvog upotrebljivog programa koji će sadržavati samo strukturne i/ili dopunske varijable, dakle rješenje u kojem će sve artificijelne varijable biti izbačene iz baze [Dobrenić, 1975, 77]. Za izbacivanje tih varijabli za ovu metodu uvodi se novi d_j redak. Taj redak sadržava vrijednosti koje su dobivene zbrajanjem koeficijenta a_{ij} okomito po stupcu za strukturne varijable i ograničenja [Dobrenić, 1975,91]. Odnosno kako je taj d_j redak vezan za upravo te artificijelne varijable i uvodimo ga kako bi ih se moglo riješiti iz baze, vrijednost unutar reda sadrži zbroj okomito vrijednosti po stupcu, ali samo za one koeficijente unutar čijeg reda se nalazi artificijelna varijabla u bazi.

Cilj prve faze 'dvofazne' metode je izbaciti artificijelne varijable iz baze. Jednom kada one izađu ne mogu se više pojaviti unutar mogućeg rješenja. Sve dok one postoje u bazi rješenje problema nije moguće. Provođenje ove faze odnosno njen algoritam glasi slično kao i za problem maksimuma. Odabrati vodeći stupac, odnosno odabrati varijablu koja ulazi u bazu te odabrati na koje mjesto u bazi ona ulazi tako da se odredi vodeći red. Za problem minimuma to se izvodi tako da se unutar d_j reda pronalazi najveća pozitivu vrijednost - ispod strukturnih i dopunskih varijabli, jer artificijelne ne smiju ulaziti u bazično rješenje – i stupac u kojem se ta vrijednost nalazi odabiramo kao vodeći. Odabir vodećeg reda odvija se na isti način kao i u standardnom problemu za maksimum, po pravilu minimalnog omjera. Ovdje se može spomenuti i još jedan način kraćenja vremena računanja, a taj je da će vrijednost ispod artificijelne varijable koja je bila vezana za negativnu dopunsku varijablu imati jednake koeficijente u tablici obrnutog predznaka, stoga kada se u tablici izračunaju elementi varijable u_4 elementi ispod varijable w_4 mogu se prepisati s obrnutim predznakom.

Druga faza započinje ispitivanjem je li ovo dobiveno prvo moguće rješenje ujedno i optimalno, tako da se u $Z_j - c_j$ redu ispitaju vrijednosti ispod nebazičnih strukturnih i dopunskih varijabli. Kako za minimum vrijednosti tog red da bi program bio optimalan moraju biti ≤ 0, odnosno ne smije postojati više niti jedna varijabla koja bi mogla ući u bazu i da dodatno umanji funkciju cilja. Ukoliko u tom redu postoje pozitivne vrijednosti znači da se trenutni program može još poboljšavati. Izbor varijable koja će ući u novo bazično rješenje vrši se po osnovnom kriteriju odnosno $\boldsymbol{max\{Z_j - c_j\}}$ gdje je $\boldsymbol{Z_j - c_j > 0.}$

4.3. Opći problemi linearnog programiranja za maksimum i minimum

Unutar realnih sustava javljaju se već spomenuti problemi koji znaju imati i po par stotina varijabli i vjerojatnost da su im ograničenja postavljena u općem obliku vrlo je velika. Jedan od općih problema bit će i riješen u sljedećem poglavlju. Opći problemi za minimum i maksimum kako je već navedeno zadani su ograničenjima koja su iskazana nejednadžbama 'manjim od ili jednako' i/ili 'većim od ili jednako' te mogu biti i u obliku jednadžbi. Svođenje na kanonski oblik unutar ovakvih problema dovodi do toga da se i u problemu za maksimum moraju uvesti artificijelne varijable, a sukladno tome i d_j redak. Postupak pronalaženja optimalnog programa isto se odvija u dvije faze – 'dvofaznom' simpleks metodom. Jedina razlika koja se javlja u općem problemu za maksimum spram standardnog i općeg problema za minimum (osim one vezane za redak $Z_j - c_j$) je ta da pošto se unutar općeg problema u kanonskom obliku uz artificijelne varijable unutar funkcije cilja morao uvest dovoljno velik odnosno mali broj M - iskazan negativnim predznakom - vrijednosti u dj retku će također biti iskazane s negativnim predznakom. Zatim se preko toga reda unutar prve faze odabirom $\boldsymbol{max\{|d_j|\}}$ gdje je $d_j < 0$ određuje vodeći stupac. I to je jedina važna nova razlika koja se treba spomenuti. Test programa je li on optimalan unutar druge faze 'dvofazne' metode odvija se na jednak način kao i unutar standardnog problema za maksimum. Transformacije vodećeg reda i novih elemenata odvijaju se također po istoj formuli.

5. Posebna svojstva simpleks algoritma

Problemi linearnog programiranja nisu uvijek 'idealni' odnosno postupak rješavanja ne teče uvijek kao 'školski primjerak' pa da se jednoznačno dođe do optimalnog rješenja. Prilikom rješavanja problema simpleks metodom odnosno simpleks algoritmom moguće je da se pojave određeni problemi, nedoumice, nedopuštene vrijednosti i sl. Kako se pristupa takvim problemima i može li ih se i na koji način riješiti opisat će se u nastavku poglavlja. Prije opisivanja svakog od posebnih svojstava koja se javljaju u kratko će se detaljnije opisati pojam bazičnog rješenja nego što se spominjao u ostatku rada.

5.1. Bazično rješenje

Unutar ovog poglavlja malo će se više objasniti bazično rješenje kako bi se sljedeće poglavlje moglo lakše razjasniti. Bazično rješenje je rješenje koje sadrži najviše m varijabli različitih od nule, dok su ostale varijable jednake nula [Vučković, 1983, 81]. Ako su te varijable koje su različite od nule ujedno sve i pozitivne to rješenje zove se **bazično moguće rješenje**. Upravo zato kada se uvode artificijelne varijable u kanonskom problemu želi se izbjeći rješenje s negativnim varijablama unutar prvog postavljenog rješenje jer takvo rješenje nije prihvatljivo kao moguće odnosno takvo rješenje s negativnim vrijednostima je **nemoguće bazično rješenje**. Ukoliko je bazično moguće rješenje ujedno i najbolje moguće bazično rješenje u smislu oblika optimalnosti koja se želi postići nad problemom to rješenje naziva se **optimalnim rješenjem problema.**

Unutar mogućih bazičnih rješenja javlja se pojam degeneriranog rješenja. Ukoliko se unutar mogućeg bazičnog rješenja pojavi barem jedna varijabla s vrijednosti 0 onda se takvo rješenje naziva **bazično degenerirano moguće rješenje** [Babić, 2010, 37]**.** A ukoliko su sve varijable unutar baze, točno njih m, pozitivne takvo rješenje zove se **bazično nedegenerirano moguće rješenje**.

5.2. Degeneracija

Što znači degenerirano bazično rješenje i kako se ono prepoznaje opisano je u prethodnom potpoglavlju. Degeneracija unutar simpleks tabele primjećuje se prilikom provođenja pravila minimalnog omjera Θ. Ona se javlja kada se ne može jednoznačno odrediti koji redak postaje vodeći redak odnosno koju bazičnu varijablu treba izbaciti iz baze. To se dogodi kada je za dvije odnosno maksimalno m bazičnih varijabli jednaka vrijednost minimalnog omjera Θ tj. kvocijenta preko kojeg se određuje vodeći red. Znači da minimum

omjera nije jedinstven. Sljedeći primjer koji će biti zadan tekstualno odnosno u okvirima ekonomskog pristupa pokazat će pojavu upravo ovog opisanog problema:

Primjer 3[9]:

Poduzeće želi proizvesti proizvode **P_1, P_2, P_3 i P_4** uz maksimalizaciju dobiti. Istraživanja proizvodnog procesa pokazala su da se ograničenja u tom procesu nalaze na strani jednog **mjesta ručne dorade**, **jedne vrste materijala** i **jedne grupe strojeva**. Grupa radnih mjesta raspolaže s **minimalno 1000 radnih sati** dok je **raspoloživa količina materijala 2000 težinskih jedinica**. **Kapacitet grupe strojeva je 2500 sati rada** i **zadan je gornjom granicom**. Potrebna količina materijala po jedinici proizvoda iznosi **4, 2, 1 i 4** težinske jedinice. Vrijeme ručne dorade po jedinici proizvoda na grupi radnih mjesta iznosi **3, 2, 2 i 3** vremenske jedinice rada. Potrebna vremena izrade pojedinih proizvoda na grupi strojeva iznosi **5, 3, 2 i 3** vremenske jedinice. Pored ovih proizvodnih ograničenja zadano je i ograničenje koje opisuje **specijalnu završnu strojnu obradu proizvoda**. **Raspoloživa količina** ovog resursa je **1000 sati rada** i **mora biti u potpunosti iskorištena**. Potrebna vremena specijalne završne obrade pojedinih proizvoda su: **1, 2, 2 i 0** vremenskih jedinica. U programskom razdoblju **planirani prihod** po jedinici proizvoda iznosi: **20, 10, 35 i 45 novčanih jedinica.**

Ključne pojmove odnosno vrijednosti označilo se u tekstu. Proizvodi **P_1, P_2, P_3 i P_4** predstavljaju proizvode koji se proizvode na tri radna mjesta: mjesto ručne dorade, jedna grupa strojeva i specijalna završna strojna obrada. Ta radna mjesto uz raspoloživu količinu materijala čine ograničenja ovog problema. Prevede li se ovaj tekst u simbolički algebarski prikaz na način on poprima sljedeći oblik:

Planirani prihod $\boldsymbol{Z = 20x_1 + 10x_2 + 35x_3 + 45x_4 \rightarrow max}$

Mjesto ručne dorade $3x_1 + 2x_2 + 2x_3 + 3x_4 \geq 1000$

Materijal $4x_1 + 2x_2 + 1x_3 + 4x_4 \leq 2000$

Grupa strojeva $5x_1 + 3x_2 + 2x_3 + 3x_4 \leq 2500$

Specijalna obrada $1x_1 + 2x_2 + 2x_3 + 0x_4 = 1000$

$x_1, x_2, x_3, x_4 \geq 0$

Pripadni kanonski problem je:

[9] Zadatak preuzet iz materijala dostupnih na e-kolegiju 'Operacijska istraživanja 1' – riješen samostalno

$$\boldsymbol{Z = 20x_1 + 10x_2 + 35x_3 + 45x_4 + 0(-u_1 + u_2 + u_3) - M(w_1 + w_2) \rightarrow max}$$

$$\begin{aligned} 3x_1 + 2x_2 + 2x_3 + 3x_4 - \boldsymbol{u_1} \quad + \boldsymbol{w_1} \quad &= 1000 \\ 4x_1 + 2x_2 + 1x_3 + 4x_4 \quad + \boldsymbol{u_2} \quad &= 2000 \\ 5x_1 + 3x_2 + 2x_3 + 3x_4 \quad + \boldsymbol{u_3} \quad &= 2500 \\ 1x_1 + 2x_2 + 2x_3 + 0x_4 \quad + \boldsymbol{w_2} \quad &= 1000 \end{aligned}$$

$$x_1, x_2, x_3, x_4, \boldsymbol{u_1, u_2, u_3, w_1, w_2} \geq 0$$

Početna tablica ovog problema sada izgleda ovako:

Tablica 14. Početna tablica za primjer degeneracije.

$c_j \rightarrow$			20	10	35	45	0	0	0	-M	-M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable			Artificijelne varijable			
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	w_1	w_2	K	R
-M	w_1	1000	3	2	2	3	-1	0	0	1	0	1010	
0	u_2	2000	4	2	1	4	0	1	0	0	0	2012	
0	u_3	2500	5	3	2	3	0	0	1	0	0	2514	
-M	w_2	1000	1	2	2	0	0	0	0	0	1	1006	
	$z_j - c_j$	0	-20	-10	-35	-45	0	0	0	0	0	-110	
	d_j	-2000	-4	-4	-4	-3	1	0	0	-1	-1	-2016	

Prilikom odabira vodećeg stupca javljaj se problem kojeg odabrati s obzirom da tri nebazične varijable u zadnjem redu poprimaju tri iste maksimalne vrijednosti. Kako se točno taj problem naziva i zašto reći će se nešto kasnije. Za sada je dovoljno primijeniti jedno od pravila koje je ujedno i najbrže, a to je da se ukoliko se pojavi ovakav problem za izbor vodećeg stupca prelazi na red $Z_j - c_j$. U njemu se sada gleda koja od nebazičnih varijabli (koje su imale u d_j istu maksimalnu vrijednost) poprima maksimalnu apsolutnu vrijednost – pošto se radi o problemu za maksimum. Varijabla koja ima najveću tu vrijednost je varijabla x_3.

Za odabir vodećeg reda izračunati su redom sljedeći omjeri:

Tablica 15. a) Odabir vodećeg reda za prvu iteraciju za primjer degeneracije.

$c_j \rightarrow$			20	10	35	45	0	0	0	-M	-M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable			Artificijelne varijable			
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	w_1	w_2	K	R
-M	w_1	1000	3	2	2	3	-1	0	0	1	0	1010	500
0	u_2	2000	4	2	1	4	0	1	0	0	0	2012	2000
0	u_3	2500	5	3	2	3	0	0	1	0	0	2514	1250
-M	w_2	1000	1	2	2	0	0	0	0	0	1	1006	500
	$z_j - c_j$	0	-20	-10	-35	-45	0	0	0	0	0	-110	
	d_j	-2000	-4	-4	-4	-3	1	0	0	-1	-1	-2016	

Iz prikazane tablice jasno je vidljivo kako dvije bazične varijable - w_1 i w_2 – poprimaju jednake vrijednosti traženog omjera. Ovo je slučaj koji će sigurno polučiti degenerirano bazično rješenje. Jer pošto je ovim bazičnim varijablama omjer - koji određuje za koliko će se smanjiti bazična varijabla ulaskom nebazične, odnosno vrijednost za koju se može maksimalno povećati nebazična varijabla - jednak, jasno je da kada iz baze izađe jedna od njih druga će u sljedećoj iteraciji poprimiti vrijednost nula. Odnosno bazično nedegenerirano rješenje postaje degenerirano. Varijable koje su se našle u tom problemu odnosno koje su u bazičnom rješenju poprimile vrijednost nula također će kasnije kroz iteracije izaći iz bazičnog rješenja, ali te promjene neće zbog te nule utjecati na vrijednost funkcije cilja odnosno funkcija cilja ostat će nepromijenjena. Cijeli taj nepromijenjeni proces zbog nula vrijednosti zove se degeneracija. Razlog tome može se objasniti tako, pogledaju li se nejednadžbe ograničenja kao određeni vektori, da se u trenutnom bazičnom rješenju dogodio slučaj u kojem postoji moguća točka optimalnog rješenja u kojoj se nalazi više nego dva sjecišta ograničenja [Kalpić, Molnar, 1996, 31]. Pomicanje iz jednog sjecišta u drugo koje se nalazi u istoj točki, ne utječe na promjenu vrijednosti funkcije.

Različiti autori različito pristupaju ovom problemu, a svi se vode time da se u što manje koraka odnosno iteracija dođe do optimalnog rješenja. Bitno je za naglasiti kako u slučajevima kada se javlja degeneracija, i to najčešće kada je više od jednog bazičnog rješenja jednako nuli, javlja se još jedan problem, problem kruženja. Njega se također pokušava što više izbjeći uporabom različitih pristupa rješavanja problema degeneracije. On će biti više objašnjen kasnije. Jedan od načina kojim se prilazi degeneraciji je da se od više varijabli koje su vezane istim minimalnim omjerom proizvoljno odabere varijabla koja izlazi iz baze. Drugi

od načina je da iz baze izlazi ona varijabla čija je vrijednost indeksa manja odnosno ona koja se u ograničenjima u kanonskom problemu javila prva.

Način na koji će se ovdje pristupiti problemu glasi da ukoliko se pojave dvije (ili više) bazičnih varijable s ovim jednakim omjerima da će se u omjer staviti prvi sljedeći stupac koji nije ujedno i vodeći. Vrijednosti tog stupca opet se dijele vrijednostima vodećeg stupca, ali samo za vezane varijable odnosno one koje su imali jednak minimalni uvjet za izlazak iz baze. To pomicanje po stupcima odvija se toliko dugo dok se ne dobiju različite vrijednosti. U odabranom primjeru već odabirom prvog stupca dobile su se različite vrijednosti i za vodeći red odabire se četvrti red.

Tablica 16. b) Odabir vodećeg reda za prvu iteraciju za primjer degeneracije.

$c_j \rightarrow$			20	10	35	45	0	0	0	-M	-M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable			Artificijelne varijable			
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	w_1	w_2	K	R
-M	w_1	1000	3	2	2	3	-1	0	0	1	0	1010	500 (1,5)
0	u_2	2000	4	2	1	4	0	1	0	0	0	2012	2000
0	u_3	2500	5	3	2	3	0	0	1	0	0	2514	1250
-M	w_2	1000	1	2	**2**	0	0	0	0	0	1	1006	500 (0,5)
	$z_j - c_j$	0	-20	-10	-35	-45	0	0	0	0	0	-110	
	d_j	-2000	-4	-4	-4	-3	1	0	0	-1	-1	-2016	

Sljedeća iteracija pokazuje upravo spomenutu pojavu degeneracije gdje oba reda u kojoj su bile vezane bazične varijable poprimaju vrijednost 0. Pojava nule u bazičnom rješenju označava degenerirano bazično rješenje. Kako dobiveni program i dalje nije optimalan, jer je prisutna artificijelna varijabla u bazi odnosno d_j red i dalje nema sve nule, ide se dalje u sljedeću iteraciju, odnosno koraci prve faze 'dvofazne' metode i dalje se provode:

Tablica 17. Prva iteracija za primjer degeneracije.

$c_j \rightarrow$			20	10	35	45	0	0	0	-M	-M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable			Artificijelne varijable			
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	w_1	w_2	K	R
-M	w_1	0	2	0	0	**3**	-1	0	0	1	-1	4	**0**
0	u_2	1500	7/2	1	0	4	0	1	0	0	-1/2	1509	375
0	u_3	1500	4	1	0	3	0	0	1	0	-1	1508	500
35	x_3	500	1/2	1	1	0	0	0	0	0	1/2	503	/
	$z_j - c_j$	17500	-5/2	25	0	-45	0	0	0	0	35/2	17495	
	d_j	0	-2	0	0	-3	1	0	0	-1	1	-4	

Unutar sljedeće, druge iteracije, dobilo se upravo onakvo bazično rješenje kakvo je prethodno objašnjeno. Kako je vrijednost - koja pokazuje za koliko se varijabla koja ulazi u bazu može najviše promijeniti - jednaka nuli, vrijednost te nebazične varijable koja ulazi u bazu bit će naravno nula, a nova vrijednost funkcije cilja ostat će ne promijenjena. I stoga ova iteracija nije nužno povećala funkciju cilja, ali je omogućila drugačiju kombinaciju varijabli u bazi odnosno pomoću ove iteracije izbačena je neželjena artificijelna varijabla iz baze i time se došlo do kraja prve faze 'dvofazne' metode.

Tablica 18. Druga iteracija za primjer degeneracije.

$c_s \downarrow$	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	w_1	w_2	K	R
45	x_4	0	2/3	0	0	1	-1/3	0	0	1/3	-1/3	4/3	
0	u_2	1500	5/6	1	0	0	4/3	1	0	-4/3	5/6	4511/3	
0	u_3	1500	2	1	0	0	1	0	1	-1	0	1504	
35	x_3	500	1/2	1	1	0	0	0	0	0	1/2	503	
	$z_j - c_j$	17500	55/2	25	0	0	-15	0	0	15	5/2		

Drugom iteracijom i dalje se nije postigao željeni optimum programa što znači da se ulazi u drugu fazu metode i odabirom maksimalne negativne vrijednosti u zadnjem redu odabire se vodeći stupac te zatim i vodeći red.

Tablica 19. Odabir vodećeg reda i stupca za treću iteraciju za primjer degeneracije.

$c_j \rightarrow$			20	10	35	45	0	0	0	-M	-M		
c_s ↓	Bazično rješenje		Strukturne varijable				Dopunske varijable			Artificijelne varijable			
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	w_1	w_2	K	R
45	x_4	0	2/3	0	0	1	-1/3	0	0	1/3	-1/3	4/3	/
0	u_2	1500	5/6	1	0	0	**4/3**	1	0	-4/3	5/6	4511/3	**1125**
0	u_3	1500	2	1	0	0	1	0	1	-1	0	1504	1500
35	x_3	500	1/2	1	1	0	0	0	0	0	1/2	503	/
	$z_j - c_j$	17500	55/2	25	0	0	-15	0	0	15	5/2		

Nakon provedenih transformacija dobiva se sljedeća ujedno i posljednja iteracija programa:

Tablica 20. Treća iteracija za primjer degeneracije.

c_s ↓	Bazično rješenje		Strukturne varijable				Dopunske varijable			Artificijelne varijable			
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	w_1	w_2	K	R
45	x_4	375	7/8	1/4	0	1	0	1/4	0	0	-1/8	1509/4	
0	u_1	1125	5/8	3/4	0	0	1	3/4	0	-1	5/8	5411/4	
0	u_3	375	11/8	1/4	0	0	0	-3/4	1	0	-5/8	1505/4	
35	x_3	500	1/2	1	1	0	0	0	0	0	1/2	503	
	$z_j - c_j$	34375	295/8	145/4	0	0	0	45/4	0	0	95/8	137885/4	

Iz navedenog optimalnog programa može se vidjeti kako se dobilo bazično nedegenerirano rješenje. Stoga se može zaključiti da pojava degeneracije unutar simpleks algoritma ne mora nužno značiti da se na kraju neće doći do nedegeneriranog rješenja odnosno i do optimalnog rješenja. Znači samo da će možda prilikom provođena algoritma biti potrebna koja iteracija više kako bi se došlo do konačne optimalne kombinacije bazičnih varijabli. Objašnjena degeneracija još se može nazivati i **primarna degeneracija**, a degeneracija koja će se sljedeće opisati naziva se **dualna degeneracija.**

5.2.1. Dualna degeneracija

Dualna degeneracija javlja se u simpleks tablici kada se unutar kriterija optimalnosti jave dvije jednake vrijednosti. Točnije, kada se unutar izbora za vodeći stupac tj. strukturne nebazične varijable koja treba ući u bazu pojavi više 'kandidata'. Naziva se dualna

degeneracija iz razloga što se ispod strukturnih varijabli u konačnom rješenju problema u zadnjem redu čitaju vrijednosti dopunskih varijabli dualnog problema.

U prijašnjem primjeru pojavio se takav slučaj (i pristupilo mu se tako da se na najbrži mogući način odabrala varijabla koja treba ući u bazu. Odlučilo se na temelju reda relativnih troškova. Još jedan od načina pristupanja ovom problemu je da se obični kriterij izbora nebazične varijable zamjerni s egzaktnim. Odnosno da se od mogućih varijabli koje bi trebale ući u bazu vidi koja će dati pogodniji relativni trošak za sljedeću iteraciju. Sljedeći primjer[10] koji će biti zadan odmah prvom početnom simpleks tablicom prikazat će na koji se način koristi taj egzaktni sporiji, ali točniji kriterij izbora nebazičnih varijabli. Treba još jednom naglasiti samo kako ni jedan izbor varijable u ovom problemu nije pogrešan, samo će se taj izbor odraziti na ukupan broj iteracija. Ukoliko se izabere jedna varijabla koja možda nije trebala ući, već u nekim od sljedećih iteracija ona varijabla koja je trebala ući odabrat će se za vodeći stupac.

Tablica 21. Početna tablica za primjer dualne degeneracije.

$c_j \rightarrow$			20	30	45	40	0	0	0	0	M	M	M	M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				Artificijelne varijable					
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	u_4	w_1	w_2	w_3	w_4	K	R
M	w_1	500	2	3	3	1	-1	0	0	0	1	0	0	0	509	500/3
M	w_2	1500	1	4	5	0	0	-1	0	0	0	1	0	0	1510	300
M	w_3	1500	3	3	1	1	0	0	-1	0	0	0	1	0	1508	1500
M	w_4	2000	0	0	4	4	0	0	0	-1	0	0	0	1	2008	500
	$z_j - c_j$	0	-20	-30	-45	-40	0	0	0	0	0	0	0	0	-135	
	d_j	5500	6	10	13	6	-1	-1	-1	-1	1	1	1	1	5535	

Sljedeće iteracije bit će prikazane jedna za drugom dok se ne pojavi ona u kojoj se javlja dualni oblik degeneracije.

[10] Zadatak preuzet iz materijala dostupnih na e-kolegiju 'Operacijska istraživanja 1' – riješen samostalno

Tablica 22. Prva za primjer dualne degeneracije.

$c_j \rightarrow$			20	30	45	40	0	0	0	0	M	M	M	M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				Artificijelne varijable					
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	u_4	w_1	w_2	w_3	w_4	K	R
45	x_3	500/3	2/3	1	1	1/3	-1/3	0	0	0	1/3	0	0	0	509/3	/
M	w_2	2000/3	-7/3	1	0	-5/3	**5/3**	-1	0	0	-5/3	1	0	0	1985/3	**400**
M	w_3	4000/3	7/3	2	0	2/3	1/3	0	-1	0	-1/3	0	1	0	4015/3	4000
M	w_4	4000/3	-8/3	-4	0	8/3	4/3	0	0	-1	-4/3	0	0	1	3988/3	1000
	$z_j - c_j$	7500	10	15	0	-25	-15	0	0	0	15	0	0	0	7500	
	d_j	10000/3	-8/3	-3	0	5/3	10/3	-1	-1	-1	-10/3	1	1	1	9988/3	

Tablica 23. Druga iteracija za primjer dualne degeneracije.

$c_j \rightarrow$			20	30	45	40	0	0	0	0	M	M	M	M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				Artificijelne varijable					
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	u_4	w_1	w_2	w_3	w_4	K	R
45	x_3	300	1/5	4/5	1	0	0	-1/5	0	0	0	1/5	0	0	302	/
0	u_1	400	-7/5	-3/5	0	-1	1	-3/5	0	0	-1	3/5	0	0	397	/
M	w_3	1200	14/5	11/5	0	1	0	1/5	-1	0	0	-1/5	1	0	1206	1200
M	w_4	800	-4/5	-16/5	0	**4**	0	4/5	0	-1	0	-4/5	0	1	800	**200**
	$z_j - c_j$	13500	-11	6	0	-40	0	-9	0	0	0	9	0	0	13455	
	d_j	2000	2	-1	0	5	0	1	-1	-1	0	-1	1	1	2006	

Tablica 24. Treća iteracija za primjer dualne degeneracije.

$c_j \rightarrow$			20	30	45	40	0	0	0	0	M	M	M	M		
$c_s \downarrow$	Bazično rješenje			Strukturne varijable			Dopunske varijable				Artificijelne varijable					
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	u_4	w_1	w_2	w_3	w_4	K	R
45	x_3	300	1/5	4/5	1	0	0	-1/5	0	0	0	1/5	0	0	302	300
0	u_1	600	-8/5	-7/5	0	0	1	-2/5	0	-1/4	-1	2/5	0	1/4	597	600
M	w_3	1000	3	3	0	0	0	0	-1	1/4	0	0	1	-1/4	1006	1000
40	x_4	200	-1/5	-4/5	0	1	0	1/5	0	-1/4	0	-1/5	0	1/4	200	200
	$z_j - c_j$	21500	-19	-26	0	0	0	-1	0	-10	0	1	0	10	21455	21500
	d_j	1000	**3**	**3**	0	0	0	0	-1	1/4	0	0	1	-1/4	1006	1000

U trećoj iteraciji dogodila se navedena degeneracija. Po običnom kriteriju gledajući relativne troškove odabrali bi onaj maksimalni među pozitivnim. U ovom slučaju zapravo prema tom redu zapravo imamo tkz. optimalno rješenje jer nema pozitivnih u $Z_j - c_j$, ali naravno da je rješenje daleko od mogućeg kamoli optimalnog jer u bazi još uvijek postoji artificijelna varijabla. Ovaj slučaj riješit će se uporabom egzaktnog kriterija. Iako se on primjenjuje kada još ima pozitivnih vrijednosti, svejedno će poslužiti za ovaj slučaj. Također uporaba egzaktnog kriterija ne mora se uvijek izvršavati kada se pojavljuje problem dualne degeneracije.

Egzaktni kriterij pomoću kriterija za izbor vodećeg reda – pravilo minimalnog omjera – gleda koja je varijabla više pogodnija za ulazak u bazu. Egzaktni kriterij dat će vrijednost koja predstavlja za koliko se egzaktno može povećati funkcija cilja. Taj kriterij množi minimalni omjer varijable s njezinim relativnim troškom i odabire onu koja ima maksimalnu tu vrijednost:

$$\boldsymbol{max\Theta \cdot (Z_j - c_j)},$$

$$\Theta = min\left\{\frac{a_{i0}}{a_{is}}\right\}.$$

Znači da za odabrani primjer vrijednosti koje su se dobile razmatranjem ovog kriterija su:

Tablica 25. Egzaktni kriterij

x_1	x_2
$\Theta = min\left\{1500, \frac{1000}{3}, \right\} = \mathbf{\frac{1000}{3}}$	$\Theta = min\left\{375, \frac{1000}{3}, \right\} = \mathbf{\frac{1000}{3}}$
$max\Theta \cdot (Z_j - c_j) = max\left\{\left(\mathbf{\frac{1000}{3}} \cdot -19\right), \left(\mathbf{\frac{1000}{3}} \cdot -26\right)\right\} = max\left\{-\frac{19000}{3}, -\frac{26000}{3}\right\} = \mathbf{-\frac{19000}{3}}$	

Maksimalna vrijednost po ovom kriteriju je ona za varijablu x_1 i prema tome ona ulazi u bazu, a varijabla koja je izašla je naravno artificijelna.

Tablica 26. Četvrta iteracija za primjer dualne degeneracije.

$c_j \rightarrow$			20	30	45	40	0	0	0	0	M	M	M	M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				Artificijelne varijable					
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	u_4	w_1	w_2	w_3	w_4	K	R
45	x_3	700/3	0	3/5	1	0	0	-1/5	1/15	-1/60	0	1/5	-1/15	1/60	3524/15	700/3
0	u_1	3400/3	0	1/5	0	0	1	-2/5	-8/15	-7/60	-1	2/5	8/15	7/60	17003/15	3400/3
20	x_1	1000/3	1	1	0	0	0	0	-1/3	1/12	0	0	1/3	-1/12	1006/3	1000/3
40	x_4	800/3	0	-3/5	0	1	0	1/5	-1/15	-7/30	0	-1/5	1/15	7/30	4006/15	800/3
	$z_j - c_j$	83500/3	0	-7	0	0	0	-1	-19/3	-101/12	0	1	19/3	101/12	83479/3	83500/3
	d_j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Iz navedene tablice vidi se kako je već prilikom prve faze 'dvofazne' metode riješen problem odnosno dobiveno rješenje je ujedno i optimalno. Pogleda li se vrijednosti funkcije cilja uvidjet će se da se njena vrijednost zapravo povećala u optimalnom rješenju, ali to se može pripisati negativnim koeficijentima u redu relativnih troškova. Izbor bilo koje od dvije varijabli pogoršao bi stanje funkcije cilja u smislu da bi njenu vrijednost povećao, a to se ne želi dobiti u ovom problemu jer se njena vrijednost želi smanjiti što je više moguće. I stoga je ovdje izbor 'pao' na onu varijablu koja će manje pogoršati novo moguće rješenje. Koje je se na kraju pokazalo i optimalnim. Da se u prijašnjoj iteraciji ipak proizvoljno odabrala varijabla x_2 ne bi se dobilo pogrešno rješenje, ali morala bi se provesti jedna iteracija više s kojom bi se došlo do gore prikazanog optimalnog rješenja.

5.3. Pojava ciklusa

Ukoliko se degeneracija ne može izbjeći unutar konačnog niza iteracija odnosno novim iteracijama ne događa se stvarno poboljšanje funkcije cilja, javlja se spomenuti problem kruženja ili ciklusa. Znači da svako novo rješenje koje nije ujedno i optimalno ne donosi povećanje/smanjenje vrijednosti funkcije cilja, pa iteracije ulaze u beskrajnu petlju. Neprestano ulaze i izlaze iste varijable. U realnim sustavima ovaj slučaj se vrlo rijetko događa iako se pak nerijetko događa da puno praktičnih problema ima degenerirana rješenja [Ravindran, Phillips, Solberg, 2007, 45]. Drugim riječima moguće je da simpleks metoda unutar više iteracija ne donosi poboljšanje funkcije cilja, ali na kraju ona će izaći iz petlje i doseći optimalno rješenje.

Sljedeći primjer[11] prikazat će kako ovaj problem izgleda kada se može izaći iz petlje:

[11] Izvor: Dantzig, 1963, 230.

Tablica 27. Početna tablica za primjer kruženja.

$c_j \rightarrow$			3/4	-150	1/50	-6	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1	0	1/4	-60	-1/25	9	1	0	0	-4979/100	0
0	u_2	0	1/2	-90	-1/50	3	0	1	0	-2138/25	0
0	u_3	1	0	0	1	0	0	0	1	3	/
	$z_j - c_j$	0	-3/4	150	-1/50	6	0	0	0	15523/100	

Tablica 28. Prva iteracija za primjer kruženja.

$c_j \rightarrow$			3/4	-150	1/50	-6	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
3/4	x_1	0	1	-240	-4/25	36	4	0	0	-4979/25	/
0	u_2	0	0	30	3/50	-15	-2	1	0	703/50	0
0	u_3	1	0	0	1	0	0	0	1	3	/
	$z_j - c_j$	0	0	-30	-7/50	33	3	0	0	293/50	

Tablica 29. Druga iteracija za primjer kruženja.

$c_j \rightarrow$			3/4	-150	1/50	-6	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
3/4	x_1	0	1	0	8/25	-84	-12	8	0	-2167/25	0
-150	x_2	0	0	1	1/500	-1/2	-1/15	1/30	0	703/1500	0
0	u_3	1	0	0	1	0	0	0	1	3	1
	$z_j - c_j$	0	0	0	-2/25	18	1	1	0	498/25	

Tablica 30. Treća iteracija za primjer kruženja.

$c_j \rightarrow$			3/4	-150	1/50	-6	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
1/50	x_3	0	25/8	0	1	-525/2	-75/2	25	0	-2167/8	/
-150	x_2	0	-1/160	1	0	**1/40**	1/120	-1/60	0	97/96	**0**
0	u_3	1	-25/8	0	0	525/2	75/2	-25	1	2191/8	2/525
	$z_j - c_j$	0	1/4	0	0	-3	-2	3	0	-7/4	

Tablica 31. Četvrta iteracija za primjer kruženja.

$c_j \rightarrow$			3/4	-150	1/50	-6	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
1/50	x_3	0	-125/2	10500	1	0	**50**	-150	0	20677/2	**0**
-6	x_4	0	-1/4	40	0	1	1/3	-2/3	0	485/12	0
0	u_3	1	125/2	-10500	0	0	-50	150	1	-20671/2	/
	$z_j - c_j$	0	-1/2	120	0	0	-1	1	0	239/2	

Tablica 32. Peta iteracija za primjer kruženja.

$c_j \rightarrow$			3/4	-150	1/50	-6	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1	0	-5/4	210	1/50	0	1	-3	0	20677/100	/
-6	x_4	0	1/6	-30	-1/150	1	0	**1/3**	0	-2138/75	**0**
0	u_3	1	0	0	1	0	0	0	1	3	/
	$z_j - c_j$	0	-7/4	330	1/50	0	0	-2	0	32627/100	

Ovdje se jasno vidi da ukoliko se u petoj iteraciji za novu iteraciju odabere vodeći stupac u_2 sljedeća šesta iteracija poprimit će jednak izgled odnosno vrijednosti kao i prva početna od koje se krenulo.

Tablica 33. Šesta iteracija za primjer kruženja.

$c_j \rightarrow$			3/4	-150	1/50	-6	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable				Dopunske varijable				
	Var	Kol	x_1	x_2	x_3	x_4	u_1	u_2	u_3	K	R
0	u_1	0	1/4	-60	-1/25	9	1	0	0	-4979/100	0
0	u_2	0	1/2	-90	-1/50	3	0	1	0	-2138/25	0
0	u_3	1	0	0	1	0	0	0	1	3	/
	$z_j - c_j$	0	-3/4	150	-1/50	6	0	0	0	15523/100	

5.4. Slučaj nedopuštenih rješenja

Slučaj nedopuštenih rješenja može se javiti već prilikom postavlja prvog početnog rješenja. Nedopušteno rješenje je ono rješenje unutar kojeg postoji barem jedna bazična varijabla koja poprima negativnu vrijednost. Upravo su se zbog toga unutar problema koji ima ograničenja/e zadana donjom granicom uvodile artificijelne varijable kako bi se već u startu izbjeglo takvo nemoguće rješenje. Proporcionalno s time ukoliko se iz bazičnog rješenja ne mogu izbaciti sve artificijelne varijable za takav problem ne postoji moguće rješenje. Grafička pozadina toga glasila bi da uopće ne postoji jedna točka rješenja tog sustava u kojoj su zadovoljeni svi uvjeti ograničenja, a kamoli da je još ta točka i optimalna. Sljedeći jednostavan primjer[12] bit će riješen prvo simpleks algoritmom (makar se ovakvi problemi ne rješavaju na taj način), a onda će njegovo rješenje biti i grafički prikazano.

$$\boldsymbol{Z = 4x_1 + 4x_2 \rightarrow min}$$

$$2x_1 - 2x_2 = 4$$
$$3x_1 + 3x_2 \geq 9$$
$$-4x_1 + 2x_2 \leq 8$$
$$4x_1 - 8x_2 \leq 3$$
$$-3x_1 + 3x_2 = 9$$
$$9x_1 + 9x_2 \leq 81$$

$$\boldsymbol{x_1, x_2 \geq 0}$$

Pripadni kanonski problem glasi:

[12] Zadatak preuzet iz materijala dostupnih na e-kolegiju 'Operacijska istraživanja 1' – riješen samostalno

$$\boldsymbol{Z = 4x_1 + 4x_2 + 0(-u_1 + u_2 + u_3 + u_4) + M(w_1 + w_2 + w_3) \rightarrow min}$$

$$\begin{aligned}
2x_1 - 2x_2 \qquad\qquad + \boldsymbol{w_1} \qquad &= 4 \\
3x_1 + 3x_2 - \boldsymbol{u_1} \qquad\qquad + \boldsymbol{w_2} \qquad &= 9 \\
-4x_1 + 2x_2 \quad + \boldsymbol{u_2} \qquad\qquad &= 8 \\
4x_1 - 8x_2 \qquad + \boldsymbol{u_3} \qquad\qquad &= 3 \\
-3x_1 + 3x_2 \qquad\qquad + \boldsymbol{w_3} \quad &= 9 \\
9x_1 + 9x_2 \qquad\qquad + \boldsymbol{u_4} \qquad &= 81
\end{aligned}$$

$$x_1, x_2 \boldsymbol{u_1, u_2, u_3, u_4, w_1, w_2, w_3} \geq \boldsymbol{0}$$

Tablica 34. Početna tablica za primjer nedopuštenog rješenja.

$c_j \rightarrow$			4	4	0	0	0	0	M	M	M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable		Dopunske varijable				Artificijelne varijable				
	Var	Kol	x_1	x_2	u_1	u_2	u_3	u_4	w_1	w_2	w_3	K	R
M	w_1	4	2	-2	0	0	0	0	1	0	0	5	/
M	w_2	9	3	3	-1	0	0	0	0	1	0	15	3
0	u_2	8	-4	2	0	1	0	0	0	0	0	7	4
0	u_3	3	4	-8	0	0	1	0	0	0	0	0	/
M	w_3	9	-3	3	0	0	0	0	0	0	1	10	3
0	u_4	81	9	9	0	0	0	1	0	0	0	100	9
	$z_j - c_j$	0	-4	-4	0	0	0	0	0	0	0	-8	
	d_j	22	2	4	-1	0	0	0	1	1	1	30	

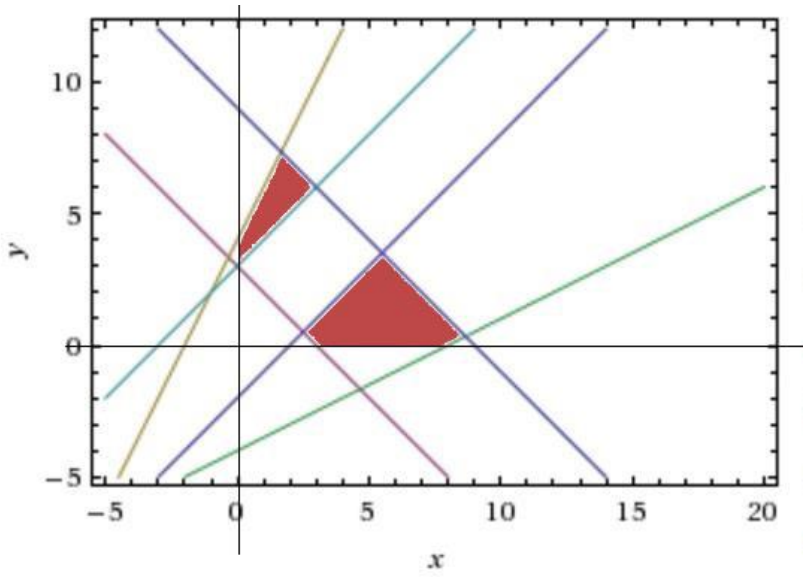
Tablica 35. Prva iteracija za primjer nedopuštenog rješenja.

$c_j \rightarrow$			4	4	0	0	0	0	M	M	M		
c_s ↓	Bazično rješenje		Strukturne varijable		Dopunske varijable				Artificijelne varijable				
	Var	Kol	x_1	x_2	u_1	u_2	u_3	u_4	w_1	w_2	w_3	K	R
M	w_1	10	4	0	-2/3	0	0	0	1	2/3	0	15	/
4	x_2	3	1	1	-1/3	0	0	0	0	1/3	0	5	/
0	u_2	2	-6	0	2/3	1	0	0	0	-2/3	0	-3	3
0	u_3	27	12	0	-8/3	0	1	0	0	8/3	0	40	/
M	w_3	0	-6	0	1	0	0	0	0	-1	1	-5	0
0	u_4	54	0	0	3	0	0	1	0	-3	0	55	18
	$z_j - c_j$	12	0	0	-4/3	0	0	0	0	4/3	0	12	
	d_j	10	-2	0	1/3	0	0	0	1	-1/3	1	10	

Tablica 36. Druga iteracija za primjer nedopuštenog rješenja.

$c_j \rightarrow$			4	4	0	0	0	0	M	M	M		
c_s ↓	Bazično rješenje		Strukturne varijable		Dopunske varijable				Artificijelne varijable				
	Var	Kol	x_1	x_2	u_1	u_2	u_3	u_4	w_1	w_2	w_3	K	R
M	w_1	10	0	0	0	0	0	0	1	0	2/3	35/3	
4	x_2	3	-1	1	0	0	0	0	0	0	1/3	10/3	
0	u_2	2	-2	0	0	1	0	0	0	0	-2/3	1/3	
0	u_3	27	-4	0	0	0	1	0	0	0	8/3	80/3	
0	u_1	0	-6	0	1	0	0	0	0	-1	1	-5	
0	u_4	54	18	0	0	0	0	1	0	0	-3	70	
	$z_j - c_j$	12	0	0	0	0	0	0	0	4/3	0	12	
	d_j	10	-2	0	0	0	0	0	1	-1/3	1	10	

Vrlo je jasno vidljivo kako je ovo nedopušteno bazično rješenje. Odnosno ovo je rješenje kojeg je nemoguće provest do kraja prve faze 'dvofazne' simpleks metode. Pošto se dalje ne može izabrati vodeći stupac, unutar baze nemoguće je riješiti se artificijelnih varijabli. Sukladno tome prikazana druga iteracija ovog primjera ne može biti moguće bazično rješenje niti može dovesti ovaj problem u prvo moguće rješenje. Stoga za ovaj problem ne postoji optimalno odnosno ni moguće rješenje.



Slika 2. Grafički[13] prikaz rješenja za primjer nedopuštenog rješenja.

5.5. Slučaj neograničenih rješenja

Ovakav slučaj unutar simpleks metode odnosno samog linearnog programiranja javlja se kada se ne može jednoznačno odrediti optimalni program. Moguće rješenje može postojati, ali ono zasigurno nije optimalno jer postoji još beskrajno mnogo rješenja koja su bolja. Slučaj neograničenih rješenja unutar simpleks algoritma javit će se kada se među omjerima preko kojih se određuje vodeći red neće moći odabrati minimalni, a da ujedno zadovoljavaju i uvjet $\Theta \geq 0$. Odnosno kada su svi ti omjeri izraženi negativnim ili beskonačnim vrijednostima $\mp\infty$. I sljedeći primjer[14] bit će jednostavan kako bi se preko njegovog grafičkog rješenja moglo lakše pojednostaviti ovo rečeno.

$$\boldsymbol{Z = 4x_1 + 5x_2 \rightarrow max}$$

$$-x_1 + 2x_2 \leq 10$$

$$x_1 - 3x_2 \leq 3$$

$$2x_1 + 3x_2 \geq 6$$

$$x_1 \geq 1$$

$$x_2 \geq 1$$

$$\boldsymbol{x_1, x_2 \geq 0}$$

[13] Izrađen samostalno u: https://www.wolframalpha.com
[14] Zadatak preuzet iz materijala dostupnih na e-kolegiju 'Operacijska istraživanja 1' – riješen samostalno

Pripadni kanonski problem glasi:

$$\boldsymbol{Z = 4x_1 + 5x_2 + 0(u_1 + u_2 - u_3 - u_4 - u_5) - M(w_1 + w_2 + w_3) \rightarrow max}$$

$$\begin{aligned}
-x_1 + 2x_2 + \boldsymbol{u_1} &= 10 \\
x_1 - 3x_2 + \boldsymbol{u_2} &= 3 \\
2x_1 + 3x_2 - \boldsymbol{u_3} + \boldsymbol{w_1} &= 6 \\
x_1 - \boldsymbol{u_4} + \boldsymbol{w_2} &= 1 \\
x_2 - \boldsymbol{u_5} + \boldsymbol{w_3} &= 1
\end{aligned}$$

$$\boldsymbol{x_1, x_2, u_1, u_2, u_3, u_4, u_5, w_1, w_2, w_3 \geq 0}$$

Postupak simpleksa neće biti cijeli prikazan već samo zadnja, četvrta iteracija u kojoj je došlo do slučaja kada se više ne može odrediti vodeći red. U toj iteraciji moguće je pročitati to neograničeno rješenje.

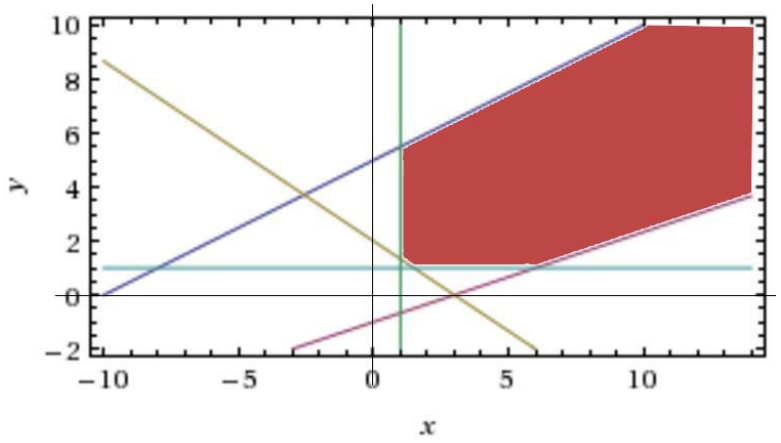
Tablica 37. Četvrta iteracija za primjer neograničenog rješenja

$c_j \rightarrow$			4	4	0	0	0		0	M	M	M		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable		Dopunske varijable					Artificijelne varijable				
	Var	Kol	x_1	x_2	u_1	u_2	u_3	u_4	u_5	w_1	w_2	w_3	K	R
0	u_3	25/2	0	0	3/2	0	1	-7/2	0	-1	7/2	0		-25/7
0	u_2	37/2	0	0	3/2	1	0	-1/2	0	0	1/2	0		-37
0	u_5	9/2	0	0	1/2	0	0	-1/2	1	0	1/2	-1		-9
4	x_1	1	1	0	0	0	0	-1	0	0	1	0		-1
5	x_2	11/2	0	1	1/2	0	0	-1/2	0	0	1/2	0		-11
	$z_j - c_j$	63/2	0	0	5/2	0	0	-13/3	0	0	13/3	0		

Kako se vidi u ovoj iteraciji sve vrijednosti omjera Θ su negativne odnosno one označavaju da odabrana ulazna nebazična varijabla može proizvoljno rasti. U realnim problemima ovakvo stanje tablice odnosno rješenja najčešće će ukazivati na pogrešno formuliran model [Kalpić, Molnar, 1996, 26].

Preko grafičkog[15] prikaza ovaj problem izgleda ovako:

[15] Izrađen samostalno u: https://www.wolframalpha.com



Slika 3. Grafički prikaz za primjer neograničenog rješenja.

Na slici se jasno vidi kako ne postoji gornje ograničenje već je područje rješenja ovog problema ide beskonačno.

5.6. Slučaj višestrukog optimalnog rješenja

Ukoliko se simpleks algoritmom dođe do optimalnog rješenja – funkcija cilja se više ne može poboljšati, a u bazi više nema artificijelnih varijabli odnosno negativnih vrijednosti – to sve ne mora značiti da je dobiveni optimalni program ujedno i jedini optimalan. Može se pojaviti slučaj takozvanog multipliciranog optimalnog rješenja odnosno slučaj u kojem postoji više optimalnih programa koji daju jednaku vrijednost zadane funkcije cilja. Ovaj slučaj može se prepoznati po tome što će nebazične varijable, odnosno varijable koje nisu ušle u rješenje ni jednog od programa, poprimiti - unutar (prvog) optimalnog rješenja - vrijednosti relativnog troška jednake nuli. Pošto su im te vrijednosti jednake nuli znači da promjena funkcije cilja unosom takvih varijabli u bazu neće biti postignuta, već će značiti da se novim programom dobilo alternativno rješenje optimalnog problema. Preko sljedećeg primjera[16] pojasnit će se ovaj slučaj.

$$\boldsymbol{Z = 3_1 + 2x_2 \rightarrow max}$$
$$-x_1 + 2x_2 \leq 4$$
$$3x_1 + 2x_2 \leq 14$$
$$x_1 - x_2 \leq 3$$
$$\boldsymbol{x_1, x_2 \geq 0}$$

[16]Izvor: Ravindran, Phillips, Solberg, 2007, 39.

Pripadni kanonski problem ovog primjera glasi:

$$\boldsymbol{Z = 3_1 + 2x_2 + 0(u_1 + u_2 + u_3) \rightarrow max}$$

$$\begin{aligned} -x_1 + 2x_2 + \boldsymbol{u_1} &= 4 \\ 3x_1 + 2x_2 + \boldsymbol{u_2} &= 14 \\ x_1 - x_2 + \boldsymbol{u_3} &= 3 \end{aligned}$$

$$\boldsymbol{x_1, x_2, u_1, u_2, u_3 \geq 0}$$

Tablica 38. Početna tablica za primjer višestrukog optimalnog rješenja.

$c_j \rightarrow$			**3**	**2**	**0**	**0**	**0**		
$c_s \downarrow$	**Bazično rješenje**		**Strukturne varijable**		**Dopunske varijable**				
	Var	**Kol**	**x_1**	**x_4**	**u_1**	**u_2**	**u_3**	**K**	**R**
0	**u_1**	**4**	-1	2	1	0	0	6	/
0	**u_2**	**14**	3	2	0	1	0	20	14/3
0	**u_3**	**3**	**1**	-1	0	0	1	4	**3**
	$z_j - c_j$	**0**	-3	-2	0	0	0	-5	

Tablica 39. Prva iteracija za primjer višestrukog optimalnog rješenja.

$c_j \rightarrow$			**3**	**2**	**0**	**0**	**0**		
$c_s \downarrow$	**Bazično rješenje**		**Strukturne varijable**		**Dopunske varijable**				
	Var	**Kol**	**x_1**	**x_4**	**u_1**	**u_2**	**u_3**	**K**	**R**
0	**u_1**	**7**	0	1	1	0	1	10	7
0	**u_2**	**5**	0	**5**	0	1	-3	8	**1**
3	**x_1**	**3**	1	-1	0	0	1	4	/
	$z_j - c_j$	**9**	0	-5	0	0	3	7	

Sljedeća ujedno i posljednja iteracija koja mijenja vrijednost funkciji cilja je:

Tablica 40. Druga iteracija za primjer višestrukog optimalnog rješenja.

$c_j \rightarrow$			3	2	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable		Dopunske varijable				
	Var	Kol	x_1	x_4	u_1	u_2	u_3	K	R
0	u_1	6	0	0	1	-1/5	8/5	42/5	
2	x_2	1	0	1	0	1/5	-3/5	8/5	
3	x_1	4	1	0	0	1/5	2/5	28/5	
	$z_j - c_j$	14	0	0	0	-1	0		

Pogleda li se bolje upravo u ovoj iteraciji optimalnog programa prisutna je nebazična varijabla koja u zadnjem redu poprima vrijednost nula. Unosom te varijable u bazu dobit će se prvo alternativno rješenje optimalnog programa ujedno i jedino. Varijabla koja izlazi van je varijabla u_1.

Tablica 41. Prvo alternativno rješenje optimalnog programa.

$c_j \rightarrow$			3	2	0	0	0		
$c_s \downarrow$	Bazično rješenje		Strukturne varijable		Dopunske varijable				
	Var	Kol	x_1	x_4	u_1	u_2	u_3	K	R
0	u_3	15/4	0	0	5/8	-1/8	0	15/4	
2	x_2	13/4	0	1	3/8	1/8	0	13/4	
3	x_1	5/2	1	0	-1/4	1/4	1	5/2	
	$z_j - c_j$	14	0	0	0	-1	0	14	

6. Zaključak

Ovaj rad pokušao je što jednostavnije približiti i objasniti način provođenja simpleks algoritma. Kako u različitim poduzećima odnosno organizacijama ne teče sve uvijek u najboljem mogućem tijeku i s najboljim mogućim ishodima, tako niti kod simpleks algoritma koji radi na temelju modela nekog problema neke organizacije nije sve uvijek 'bajno. Problemi koji se mogu javiti mogu biti dio krivo protumačenih podataka i loše postavljenog modela, no oni mogu biti i rezultat krivog provođenja nekih od metoda. Simpleks metoda nije uvijek najbolji slučaj za linearno programiranje no ovaj rad se nije time bavio. Problemi koji se mogu javiti unutar simpleks algoritma nisu nužno uvijek problemi, u punom smislu te riječi. Kako se prikazalo kod zadnjeg posebnog slučaja, slučaja višestrukog optimalnog problema, nije svako odstupanje od 'standarda' ujedno i problem. Višestruko optimalno rješenje samo pokazuje različit razmještaj varijabli unutar bazičnog rješenja koji na kraju donosi istu vrijednost funkcije cilja. Uz ovaj slučaj i ostali prikazani slučajevi javljaju se zapravo vrlo često unutar primjene simpleks algoritma i zato im je bitno pristupiti sa znanjem i logikom. Dobar dio tih slučajeva može se na neki od prikazanih načina i ukloniti odnosno protumačiti na način da se u nekom od koraka provođenja linearnog programiranja nad problem možda i pogriješilo. Simpleks algoritam zato omogućuje da se ti posebni slučajevi vrlo jasno i uoče.

Literatura

[1] Babić Z (2010) *Linearno programiranje.* Split: Sveučilište u Splitu, Ekonomski fakultet.

[2] Barković D (2001) *Operacijska istraživanja* (Drugo izmjenjeno i dopunjeno izdanje). Osijek: Grafika.

[3] Čaklović L (2010) *Geometrija linearnog programiranja* (1. Izdanje). Zagreb: Element.

[4] Dantzig G.B (1963) *Linear programming and extensions*. Princeton, New Jersey: Princeton university press.

[5] Dobrenić S (1966) *Linearno programiranje i njegova primjena u privrednoj organizaciji.* Zagreb: Informator.

[6] Dobrenić S (1975) *Linearno programiranje.* Varaždin: Fakultet organizacije i informatike.

[7] Kalpić D, Molnar V (1996) *Operacijska istraživanja.* Zagreb: ZEUS.9

[8] Lončar I, Hunjak T (1978) *Matematičke metode.* Varaždin: Fakultet organizacije i informatike.

[9] Loomba N.P (1964) *Linear programming.* USA: McGraw-Hill, Inc.

[10] Lukač Z, Neralić L (2012) *Operacijska istraživanja* (1. izadanje). Zagreb: ELEMENT d.o.o.

[11] Martić Lj (1979) *Matematičke metode za ekonomske analize* (II. svezak . treće izdanje). Zagreb: Narodne novine.

[12] Martić Lj, Vandal A (1968) *Operativno istražvanje*. Zagreb: Informator.

[13] Ravindran A, Phillips D.T, Solberg J.J (2007) *Oprations research: principles and practice* (secon edition). New Delhi: Wiley India

[14] Vučković Ž (1983) *Linearno programiranje.* Beograd: Savremena administracija.

[15] Kuzmanović I, Sabo K. *Linearno programiranje* (Radni materijal za predavanja). Dostupno 7.1.2018. na: http://www.mathos.unios.hr/lp/Materijali/predavanje16_lp.pdf

Popis slika

Popis tablica