**SVEUČILIŠTE U ZAGREBU**

**FAKULTET ORGANIZACIJE I INFORMATIKE**

**VARAŽDIN**

**Damir Vukres**

**Even Swaps kao mehanizam za odlučivanje o akcijama agenta**

**DIPLOMSKI RAD**

**Varaždin, 2016**

**SVEUČILIŠTE U ZAGREBU**

**FAKULTET ORGANIZACIJE I INFORMATIKE**

**VARAŽDIN**

**Damir Vukres**

**Broj indeksa: 39027/09-1**

**Smjer: Baze podataka i baze znanja**

**Even Swaps kao mehanizam za odlučivanje o akcijama agenta**

**DIPLOMSKI RAD**

**Mentor:**

Doc. dr. sc. Markus Schatten

**Varaždin, rujan 2016**

Sadržaj

[1. Uvod 3](#_Toc461783992)

[2. Primjer - strukturiranje problema 6](#_Toc461783993)

[2.1 Identificiranje i definiranje problema 6](#_Toc461783994)

[2.2 Određivanje skupa alternativnih rješenja 7](#_Toc461783995)

[2.3 Određivanje skupa kriterija 8](#_Toc461783996)

[3. EVEN SWAPS, standardna primjena 10](#_Toc461783997)

[3.1 Što je to Even swaps? 11](#_Toc461783998)

[3.2 Umjetnost razmjene 13](#_Toc461783999)

[3.2 Definiranje problema 16](#_Toc461784000)

[3.3 Aplikacija - sučelje 22](#_Toc461784001)

[3.4 Primjer Posao 24](#_Toc461784002)

[3.5 I Zaključak po primjeru 36](#_Toc461784003)

[4. EVEN SWAPS, Prolog i GA 38](#_Toc461784004)

[4.1 Konzistencija Even swapa 38](#_Toc461784005)

[4.2 Zamjene sa svođenjem na nulu 41](#_Toc461784006)

[4.3 Kaskadne zamjene 44](#_Toc461784007)

[4.4 Opća zamjena 45](#_Toc461784008)

[4.5 Eliminacija koeficijenta manjeg od jedan 46](#_Toc461784009)

[4.6 Od zamjena do optimizacije 48](#_Toc461784010)

[4.7 Ravnomjerne zamjene 48](#_Toc461784011)

[4.8 Kompenzacijska tabela - prologova baza podataka 50](#_Toc461784012)

[4.9 Genetski algoritam - generiranje kaskade 51](#_Toc461784013)

[4.10 Genetski algoritam - rezultati 55](#_Toc461784014)

[4.11 II Zaključak po primjeru 56](#_Toc461784015)

[5. EVEN SWAPS algoritmi i oblici 57](#_Toc461784016)

[5.1 Najveća kaskada 58](#_Toc461784017)

[5.2 Optimalan raspored 58](#_Toc461784018)

[5.3. Nivelacija na više: 61](#_Toc461784019)

[5.4. Promjena rasporeda nivelacijom 63](#_Toc461784020)

[5.5. Kriteriji za promjenu rasporeda 63](#_Toc461784021)

[5.6. Oblici 66](#_Toc461784022)

[5.7. III Zaključak po primjeru 68](#_Toc461784023)

[6. Even Swaps Agenti 71](#_Toc461784024)

[6.1. Agenti 71](#_Toc461784025)

[6.2. Više agentni sustavi 74](#_Toc461784026)

[6.3. Agenti pri Even swapsu 75](#_Toc461784027)

[6.4. BDI Agenti 76](#_Toc461784028)

[6.5. Blackboard 80](#_Toc461784029)

[6.6. B(Behavior)D(Decision) Agenti 82](#_Toc461784030)

[6.7. Kako odlučuje BD agent 87](#_Toc461784031)

[6.8. Primjeri 88](#_Toc461784032)

[6.8.1 Trgovački putnik 88](#_Toc461784033)

[6.8.2 F1- Hamilton i Raikkonen 91](#_Toc461784034)

[6.9. IV Zaključak po Primjeru 93](#_Toc461784035)

[7. Zaključak 95](#_Toc461784036)

[8. Prilozi 96](#_Toc461784037)

[8.3 Sortirani koeficijenti 96](#_Toc461784038)

[8.4 Sortirane vrijednosti solucija 97](#_Toc461784039)

[8.5 Sol1 98](#_Toc461784040)

[8.6 Last, middle, first 98](#_Toc461784041)

[8.7 Prolog kod 99](#_Toc461784042)

[8.8 Prolog listing baze 100](#_Toc461784043)

[8.9 Prolog listing program 101](#_Toc461784044)

[9.Napomene 105](#_Toc461784045)

[10.Reference 106](#_Toc461784046)

# Uvod

Even swaps se udomaćio u Teoriji odlučivanja kao brza i jednostavna metoda za višekriterijsko odlučivanje. Iako nije raširena kao neke druge metode, na primjer ELECTRE#1 ili AHP#2, i Even swaps je našao svoju primjenu što će se pokazati na nekoliko primjera.

Even swaps se teoretski može koristiti i za vrlo veliki broj argumenata. Iako bi to usporilo izvršavanje programa ipak nije argument za neki drugi pristup jer su današnja računala dovoljno brza i za takve situacije, no nisu ljudi. U situaciji u kojoj treba napraviti nekoliko stotina zamjena ljudski faktor bi potpuno onemogućio upotrebu.

Problem se, nadalje, pojavljuje i radi nekonzistencije even swapsa: rješenje može ovisiti o redoslijedu zamjena. Ta se činjenica neopravdano zanemaruje jer svodi metodu tek na pomoć kod subjektivne razrade problema. Even swap je standardni dio kolegija Teorije odlučivanja koji se tako ili vrlo slično zove na sveučilišnim programima informatike i matematike diljem svijeta. Autor nije naišao na osvrt na taj problem čitajući literaturu iz različitih izvora.

Ta se dva problema mogu riješiti „grubom silom“ (eng. brute force): even swaps bi sam napravio sve moguće zamjene te bi pokazao rang pojedinih solucija. To može biti ona solucija koja je najviše puta bila najbolja, međutim, kako se sama metoda osniva na numeričkim odnosima, mogla bi biti i ona solucija koja je postigla najveći rezultat.

Gruba sila ima očit nedostatak: u prvoj iteraciji ima n(n-1) mogućnosti, u drugoj (n-1)(n-2), itd…

n(n-1)(n-1)(n-2)(n-2)(n-3)…

n(n-1)(n-2)(n-3)… x (n-1)(n-2)(n-3)…

n! x n!/n =

Eksplozija kombinacija sugerira drugačiji pristup problemu.

Zahvaljujući diskretnim promjenama stanja koja ovise o prethodnom stanju, Even swap se može opisati kao nehomogena diferencna jednadžba prvog stupnja sa varijabilnim koeficijentima. Ovisno o načinu zamjena ti se koeficijenti drugačije reflektiraju na dobiveno stanje, pa su tako nastala dva osnovna „oblika“ (eng. pattern): zvijezda i kaskada. Tako su nazvani jer se u prvom slučaju veze zvjezdasto šire od jednog atributa na sve ostale dok se u drugom slučaju prethodna stanja kumulativno izgrađuju nova. Upravo je ta razlika akumulacije iz prethodnih stanja uzrok različitim rezultatima po različitim putovima. Oblika u even swaps mehanizmu ima puno i lako se međusobno kombiniraju.

Kaskada je najefikasniji oblik pa je po tome jedini kandidat za računanje najveće vrijednosti. No u tom slučaju postoje dodatna pravila na zamjene koje moraju ići linijom najveće akumulacije. U tom se slučaju Even swaps ponaša kao težinski graf, što ostavlja mogućnost da se upotrijebe alati koji se i inače u takvim slučajevima mogu upotrijebiti. U ovom radu je za tu svrhu odabran prolog.

Problem velike količine atributa i solucija, pogotovo ako se metodi pristupi u općenitom obliku (dakle da suprotan smjer ne mora biti recipročan – tada bi metoda izgubila prvi dio imena i zvala bi se samo Swaps) bi se obzirom na veliki broj kombinacija mogao rješavati Genetskim algoritmom. U ovom je radu prikazan jedan takav pokušaj koje se treba uzeti tek kao osnova za daljnji razvoj. Ograničen na recipročan suprotni smjer te na koeficijente =>1 radi pouzdano za relativno mali broj parametara. Kompletno rješenje Genetskim algoritmom za opći slučaj zamjena bi po obimu vjerojatno premašilo opseg cijelog ovog rada.

Prikazan je i algoritam konstruiranja najveće kaskade. I njemu treba dorada za granične situacije (recimo kombinacija koeficijenata neznatno većih od jedan sa velikim atributima i koeficijentima neznatno manjim od jedan) te za praktičnu primjenu razmatranja o oblicima even swapsa. I u tom bi slučaju opće rješenje bilo bitno opsežnije od ovdje prikazanoga.

Analiziramo li funkcionalnost Even swapsa nailazimo na sličnost sa konceptom BDI agenta. Svaki korak-po-korak proces se može predstaviti BDI paradigmom. Iz toga slijedi pokušaj da se razvijeni mehanizam rješavanja problema ovom metodom iskoristi i za izgradnju agenta kojemu bi se ponašanje preuzelo od BDI agenata. Ta kombinacija , uz još neke druge koncepte (ploča (eng. blackboard) razvijena na osnovu prologa) je odvela do pokušaja razvoja vlastitog pristupa, agenata specijaliziranih višekriterijsko odlučivanje.

U radu se koriste dva programa. Prvi je napravljen u skladu sa pristupom da se za vrijednost na koju se svodi irelevantni atribut uzme srednja vrijednost tog atributa u svim solucijama. Jedino se taj način spominje u literaturi. Za tehničko rješenje je osvojio drugo mjestu na natječaju za Innovation award:

<http://www.phpclasses.org/package/9305-PHP-Implement-decision-methods-using-Even-Swaps.html>

Objavljen je kao otvoreni kod i dostupan je na gornjoj adresi.

Primjeri su na adresi:

<http://evenswap.w-a-f.org/>

Drugi je razvijen sa pretpostavkom svođenja vrijednosti atributa na nulu. Razlozi su detaljno objašnjeni u radu. Program sa primjerima je dostupan na adresi:

<http://www.w-app.net/evenswap/>

Koristiti će se vrsta primjera koji je uobičajen za raspravu o ovoj metodi u Teoriji odlučivanja. Četiri zaključka po istom primjeru govore o drugačijim načinima pristupa metodi. Ona je svakako puno više od panela za predočavanje subjektivnih pogleda na problem.

# 2. Primjer - strukturiranje problema

## 2.1 Identificiranje i definiranje problema

Koju ponudu za posao prihvatiti obzirom na nekoliko više-manje standardnih kriterija je dilema koja se često nalazi kod freelancera (i kod drugih posloprimaca, naravno, no u ovom radu se prvenstveno radi o specifičnostima freelancera).

Često se isti ljudi znaju sresti na različitim projektima te tom prilikom razmijeniti iskustva u vezi sa različitim situacijama, no ipak je pristup svakoj od pojedinih situacija bitno drugačiji za svakog sudionika: ta vrsta posla je prilično subjektivno određena. Nerijetko je teško čuti dva slična stava o istom problemu, a gdje usuglasiti mišljenja o nekom dužem i kompleksnijem projektu.

No, takva razmjena informacija ipak daje nekakav uvid u inače nepoznato stanje nekog projekta. Koliko god se pojedine informacije trebaju uzeti sa rezervom, skup istovrsnih informacija iz različitih izvora može dati neku sliku projekta u trenutku odlučivanja, a ona opet zna vrlo često biti od nemale pomoći. Činjenica je da, zapravo, odluka u takvim situacijama predstavlja kompilaciju predviđenih posljedica iz tako dobivenih informacija.

Gledajući naknadno na takve odluke, ispadne da su one ponajviše ovisile o udjelu pojedine ulazne informacije u kompletno stvorenoj slici projekta o kojemu se razmišlja. Pitanje je što bi se desilo da je pojedinoj informaciji dan veći ili manji značaj, ili da je njihova interakcija drugačije vrednovana.

## 2.2 Određivanje skupa alternativnih rješenja

Skup alternativnih rješenja u ovome slučaju se sastoji od različitih poslovnih prilika. Nazvati ću ih Solucija 1 , Solucija 2 , itd.

Svaka od solucija ima svoje karakteristike o kojima će biti riječi u slijedećem poglavlju. Te je karakteristike čine manje ili više atraktivnom, no niti jedna nema izraženu prednost u svima.

**Solucija 1 (SOL1):**

Posao u Kopenhagenu, obavezna prisutnost kod poslodavca, kod u relativno dobrom stanju no premalenog obima (procjena na mjesec dana).

**Solucija 2 (SOL2):**

Posao u Grazu, obavezna prisutnost kod poslodavca, kod u dobrom stanju, procjena obima otprilike 3 mjeseca. Cijena rada vrlo malena.

**Solucija 3 (SOL3):**

Posao u Zagrebu, obavezna prisutnost kod poslodavca, procjena trajanja je mjesec dana, informacije o samom poslu dobre, cijena rada vrlo dobra.

**Solucija 4 (SOL4):**

Posao u Ljubljani, obavezna prisutnost kod poslodavca, kod u vrlo dobrom stanju no premalenog obima (procjena na mjesec dana). Poznat poslodavac, prijašnja suradnja je bila odlična.

**Solucija 5 (SOL5):**

Posao u Londonu, nije obavezna prisutnost kod poslodavca nakon određenog perioda, kod u lošem stanju i premalenog obima (procjena na mjesec dana). Informacije o samom poslu su loše kao i o poslodavcu.

**Solucija 6 (SOL6):**

Posao u Zadru, obavezna prisutnost kod poslodavca, kod u relativno dobrom stanju dobrog obima (procjena na 3 -4 mjeseca). Poznati suradnici, loša iskustva s njima.

## 2.3 Određivanje skupa kriterija

Vrlo često nije problem identificirati kriterije koji su bitni za vrednovanje solucija: oni su u principu jasni. No njihovi se međuodnosi ne moraju odmah pokazati, a pogotovo ih nije lako mjeriti.

Kompromisi kod definiranja kriterija mogu utjecati na odluku, no pođemo li od pretpostavke da zapravo razmatramo samo problem definiran tim kriterijima, lakše je promjenom kriterija definirati različite probleme nego nastojati sve ih svesti na isti „zajednički nazivnik“. Skup tih sličnih problema ipak ima zajedničke korijene, pa se ta porodica problema može svesti na različite utjecaje promjenjivih kriterija na one stabilne.

***Kriteriji:***

* **LOKACIJE**

Zagreb, Ljubljana, Zadar, Graz, Kopenhagen, London

Složeni prema poželjnosti od najpoželjnije destinacije do najmanje interesantne. Što je neki grad dalji to su troškovi puta veći. Tu su još bitni i troškovi boravka, kao i eventualne komplikacije oko formalnosti ove ili one vrste. Skala 1-9.

* **STANJE PROGRAMSKOG KODA**

Bitan kriterij kod preuzimanja nekog posla. Ovdje je to stanje prikazano skalom od 1-9.

* **TRAJANJE ANGAŽMANA**

Dobro-Loše

Komplicirano: nekada je poželjnije kraće, nekada duže uz jednaku većinu navedenih kriterija. Da ne bih uvodio nove, trajanje se ograničava na već gotov zaključak: 1 ili 0.

* **SURADNICI**

Kvaliteta suradnika zna biti bitna za posao. Od općih međuljudskih odnosa do stručnosti i mogućnosti dugotrajne koncentracije. Skala 1-9

* **UPRAVA**

Uprava mora znati što hoće te mora biti do te mjere dobre volje da im se može objasniti kako to napraviti. Klijent koji nije siguran što bi to htio ali zna da to mora imati odmah nije najbolji izbor za poslovnoga partnera. Također je važno doznati kako su se prije odnosili prema vanjskim suradnicima.

Skala 1-9

* **ODRŽAVANJE**

Gotov projekt zna često sadržavati skrivene bugove ili mane koje se ne mogu podvesti pod pogrešku ali ometaju izvršavanje rpograma. Osim ispravaka pod održavanje spadaju i manji zahvati u funkcionalnost.

Ima-Nema, dakle 1 ili 0.

* **ROK**

Ne-Da. 1-0.

Odgovara na pitanje je li angažman organiziran kroz definirane rokove dovršetka pojedinih faza, ili cijelog projekta. Čvrsta granica razvoja programa (deadline) je u principu nepoželjan.

* **INFORMACIJE**

Skala 1-9.

Odnosi se na saznanja o samome projektu. Nekada nije od važnosti, no nekada opet zna pomoći…

* **NAKNADA**

Iznos naknade, načini plaćanja te cijena prekovremenog, hitnog ili naknadnog rada su vrlo važna polazišta u svakom razgovoru za poslovni angažman.

Skala 1-9.

* **NA DALJINU**

Da-Ne, 1-0.

Odnosi se na mogućnost rada sa udaljene lokacije (eng. remote). Najpoželjnija karakteristika mogućeg angažmana čiji će se značaj očitovati kroz kompenzacijske vrijednosti.

# EVEN SWAPS, standardna primjena

## 

## 3.1 Što je to Even swaps?

Princip sličan Even swapsu se može naći kod Benjamina Franklina3, prema (Ref.#1):

„Podijelo sam na pola list papira linijom u dva stupca, te u jednom napisao sve što bi se moglo podvesti pod PRO u u drugom pod CON. Zatim tijekom tri ili četiri dana obzir sam stavio pod različitim pogledima kratke naznake različitih motiva koji mi se u različito vrijeme javljaju za ili protiv tih i takvih mjerila. Kad sam ih tako dobio sve zajedno na jednom mjestu, nastojao sam procijeniti njihove prave težine ... kad je svaku stvku promatra odvojeno i relativno, a cijela situacija leži preda mnom, mislim da mogu prosuditi bolje, i sam manje vjerojatno da ću krenuti lrivim smjerom, i zapravo sam našao veliku prednost od ove vrste jednadžbe koju se može nazvati Moral or Prudential Algebra“

Prema skripti (Ref.#2):“ Even swap je prikazivanje jednog atributa u terminima vrijednosti drugog“.

Pretpostavimo da neodlučni kupac jednog automobila dvoji između dva modela. Jedan, 10% skuplji, ima dizelski motor, drugi ima benzinski motor.

|  |  |  |
| --- | --- | --- |
|  | A1 | A2 |
| Cijena | 100000 | 110000 |
| Gorivo | 12 | 10 |

Kupac se odlučio za skuplji auto pa je upitan za razloge svoje odluke odgovorio da je to radi cijene goriva. Kako su cijene automobila otprilike 10000 puta veće od cijene litre goriva pretpostavimo da je to odnos te dvije veličine s kojime možemo dalje kalkulirati.

|  |  |  |
| --- | --- | --- |
|  | A1 | A2 |
| Cijena | 100000 | 110000 |
| Gorivo | 10 | 10 |

Idemo izjednačiti cijene goriva. Automobilu smo spustili cijenu goriva za 2 Kn pa moramo adekvatno povećati cijenu (odnos manje-veće se zadaje po prirodi zamjene, ovdje je očito i po drugom automobilu da je skuplji onaj koji manje troši):

|  |  |  |
| --- | --- | --- |
|  | A1 | A2 |
| Cijena | 120000 | 110000 |
| Gorivo | 10 | 10 |

Dakle 2 x10000 je 20000. Koliko smo na jednom mjestu spustili na drugom dižemo kako bi se opet vratili na početne proporcije.

Kako je gorivo sada isto u oba slučaja možemo ga sasvim izbaciti iz razmatranja. Odlučujemo prema cijeni koja je kod A2 10000 Kn niža.

Tako izbačeni atribut se zove irelevantni atribut jer ne učestvuje u odluci.

Možemo u početnu dilemu dodati i treći automobil te dodati još jedan atribut:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A1 | A2 | A3 |
| Cijena | 100000 | 110000 | 120000 |
| Gorivo | 12 | 10 | 9 |
| Osiguranje | 800 | 900 | 800 |

Za osiguranje vidimo da se radi o dva reda veličina većem broju od cijene goriva. Recimo dakle da je odnos prema gorivu 1:100 te da je manje veće, dakle da smanjenje cijene goriva podiže razinu osiguranja.

Opet izjednačimo cijene goriva:

## 

|  |  |  |  |
| --- | --- | --- | --- |
|  | A1 | A2 | A3 |
| Cijena | 120000 | 110000 | 110000 |
| Gorivo | 10 | 10 | 10 |
| Osiguranje | 1000 | 1100 | 1000 |

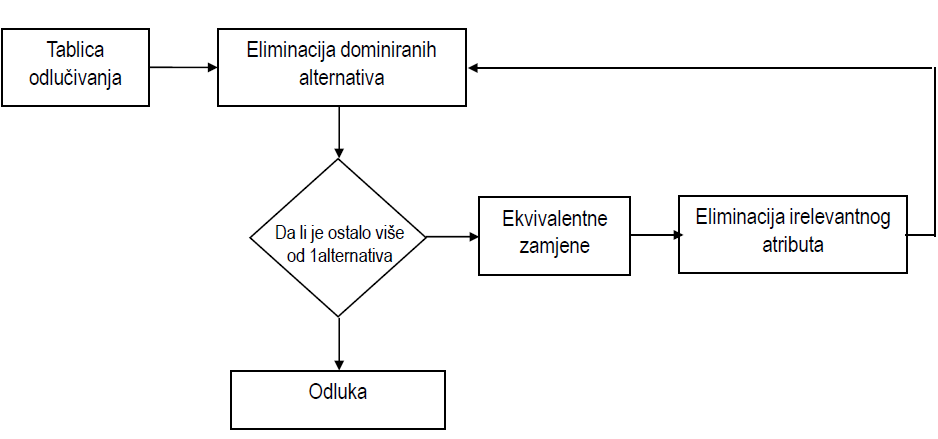
## 

Bez irelevantnog atributa:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A1 | A2 | A3 |
| Cijena | 120000 | 110000 | 110000 |
| Osiguranje | 1000 | 1100 | 1000 |

Niti jedna solucija nije najbolja po svim pitanjima. Kada bi postojala takva rekli bi da je apsolutno dominantna. Ovdje imamo praktičnu dominaciju: A3 i A2 su iste po kriteriju cijene automobila, no po kriteriju cijene osiguranja A3 je povoljniji.

Dijagram Even swapsa:



Slika 1. : Dijagram even swapsa, Izvor Ref.#2. Str.18.

Može se pratiti i navedeni primjer koji je praktički najjednostaviji mogući. Zamjene se ponavljaju dok postoji još nedominiranih solucija ili dok do zadnjeg atributa. Na kraju se odlučuje po kumuliranim vrijednostima preostalog atributa.

## 3.2 Umjetnost razmjene

„**Umjetnost razmjene**“ prema HBR-u (Ref.#3) bi trebala slijediti slijedeće savjete:

* **Prvo lagane zamjene:** situacije čije konzekvence nisu odmah jasne treba izbjegavati. Prvo u zamjenu idu najjednostavnije stvari.
* **Usredotočite se na iznos zamjene, a ne na važnosti cjelokupnog cilja.** Nema smisla reći da je jedan cilj važniji od drugoga bez obzira na stvarni stupanj posljedica za promatrane alternative. Radije treba pažnju obratiti na iznose, oni možda mogu biti značajniji.
* **Treba stalno biti svjestan da vrijednost inkrementalnih promjena ovisi o onome s čime ste počeli zamjene.** Nije dovoljno razmišljati samo o veličini jednog dijela, već se treba uzeti u obzir i veličina cijelog komada
* **Treba raditi konzistentne zamjene.** Ako bi zamijenili A za B te B za C, treba biti spreman zamijeniti i A za C.
* **Orijentirati se na jasne informacije.** Zamjene su subjektivne, no treba voditi računa i da budu, koliko je moguće, poduprte i solidnim temeljem razumijevanja materije te jasnom analizom.

Sve gore navedeno su općeniti savjeti koji zapravo ne izlaze iz okvira iste te subjektivnosti. Drugim riječima ne pomažu. Treba se vratiti na njih nakon objašnjenja oblika (pattern) Even swapa.

Iako vrlo često nije jasno na prvi pogled, princip „Quid pro Quo“4 je dio mnogih životnih situacija. Vrlo često nam se zapravo događa da razmišljamo o jednoj stvari/pojavi u terminima druge, možda povoljnije ili lakše dobavljive, ali nedvojbeno u tom trenutku zamjenjive sa drugom solucijom koja može izgledati kao problem, ili jednostavno čija realizacija izgleda teže.

Biti će jasnije kako se uđe dublje u primjere.

Prema (Ref.4) trebalo bi krenuti prvo od najlakše zamjene. Pravo bi pitanje bilo kako znati koja je zamjena laka, ili najlakša.

Diplomski rad **Josipa Tominca** s Agronomskg fakulteta (Ref.5) se bavi proizvodnjom šljiva. Odluke u tom procesu donosi uz pomoć Even swapsa. Citat:

*I korak*

*Raspoložive ponude ocjenjujmo na osnovi pet kriterija. Kriteriji na kojima ću temeljiti odluku su : dobit, koeficijent ekonomičnosti, investicijske potrebe, količina rada i tržišni rizik kojeg sam subjektivno ocijenio ocijenama od 1 do 5 (najmanji tržišni rizik ocijena 1....izrazito vielik tržišni rizik ocijena 5).*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***OPCIJE*** | *Dobit* | *Koef.ekonomičnosti* | *Investicijske potrebe* | *Rad* | *Tržišni rizik* |
| *Šljiva* | *27 273 kn* | *1,56* | *120 000 kn* | *180 h* | *1* |
| *Šljivovica* | *74 500 kn* | *1,75* | *390 000 kn* | *260 h* | *3* |
| *Pekmez* | *115 000 kn* | *2,12* | *312 000 kn* | *300 h* | *4* |

*Gledajući tabelu opcija i kriterija postavljamo si pitanje : Koju opciju bi izabrali Even Swaps metodom? Prvi korak je taj da vidimo koju opciju možemo izbaciti na temelju činjenice da je dominantna nad drugima. U našem primjeru nemamo takav slučaj.*

*II korak*

*U drugom koraku nalazimo kriterij po kojemu se opcije najmanje razlikuju. Odabrali smo koeficijent ekonomičnosti te određujemo promjenu koju bi trebali napraviti da eliminiramo kriterij “koeficijent ekonomičnosti” kao nevažan u izboru. Zatim biramo jedan od preostalih atributa (na primjer “dobit”) i procjenjujemo koliko bi trebala biti promjena dobiti koja bi kompenzirala smanjenje koeficijenta ekonomičnosti.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***OPCIJE*** | *Dobit* | *Koef.ekonomičnosti* | *Investicijske potrebe* | *Rad* | *Tržišni rizik* |
| *Šljiva* | *27 273 kn* | *1,56* | *120 000 kn* | *180 h* | *1* |
| *Šljivovica* | *67 050 kn* | *1,56* | *390 000 kn* | *260 h* | *3* |
| *Pekmez* | *85 100 kn* | *1,56* | *312 000 kn* | *300 h* | *4* |

*Ekvivalentnim opisom opcija eliminirali smo razlike između opcija po kriteriju “koeficijent ekonomičnosti”, pa ga možemo eliminirati iz daljnje analize.*

Nivelacija kompenzacijskog atributa se odvija otprilike u postotku promjene irelevantnog atributa, no to nije konkretno definirano. Autor kaže kako „…procjenjuje kolika bi trebala biti promjena dobiti…“ bez argumentacije.

Prevaga subjektivnog nad kompenzacijskom tabelom u ovom slučaju ne mijenja rezultat no ostavlja dojam kako je odluka metode više proizvoljna nego matematički jasno određena.

## 3.2 Definiranje problema

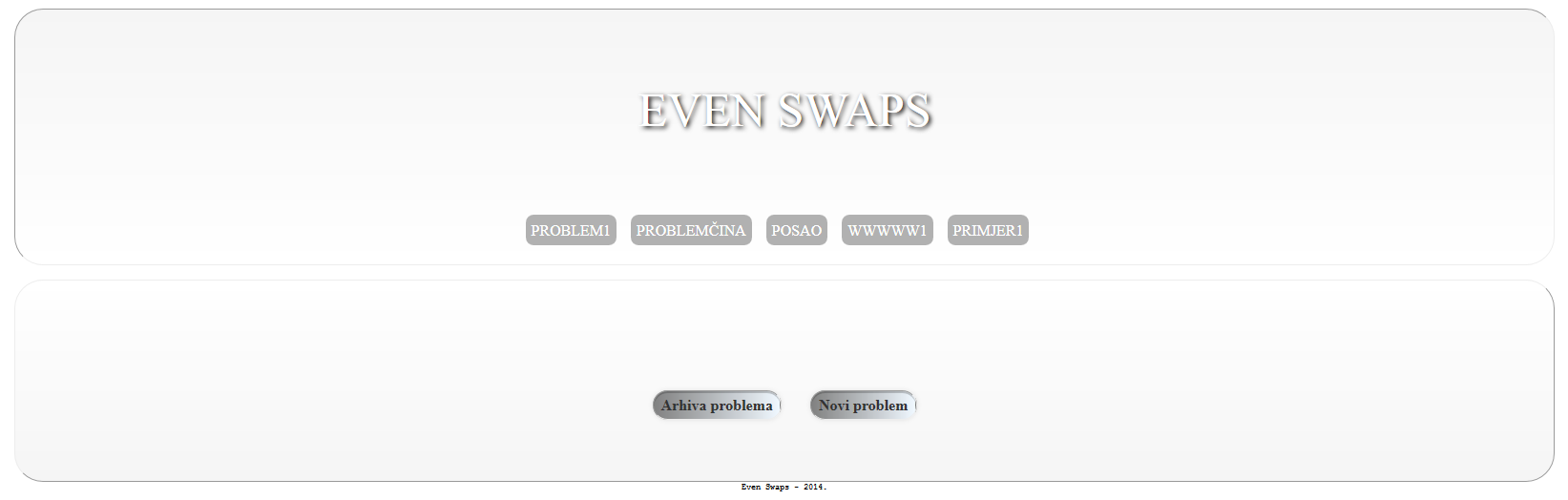
Software je dizajniran da radi sa problemima, otuda i nazivlje:

****

Slika 2. ES software

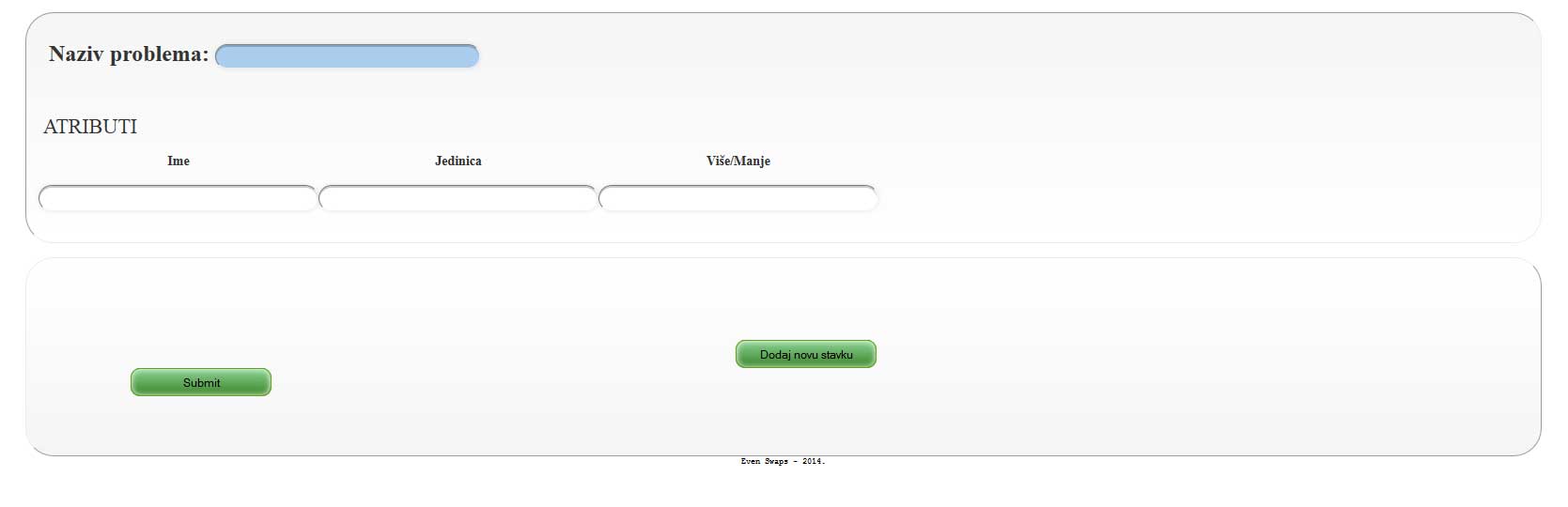
* ***Arhiva problema*** se odnosi na spremljene situacije. **Posao** je primjer iz ovoga rada, a **Primjer1** je uzet iz skripte (Ref.2) .
* **Novi problem** omogućuje kreiranje nove situacije.

Klikom na ime problema odlazi se na već definirane probleme:



Slika 3. ES software - Arhiva problema

**Novi problem** se odnosi na definiranje problema:

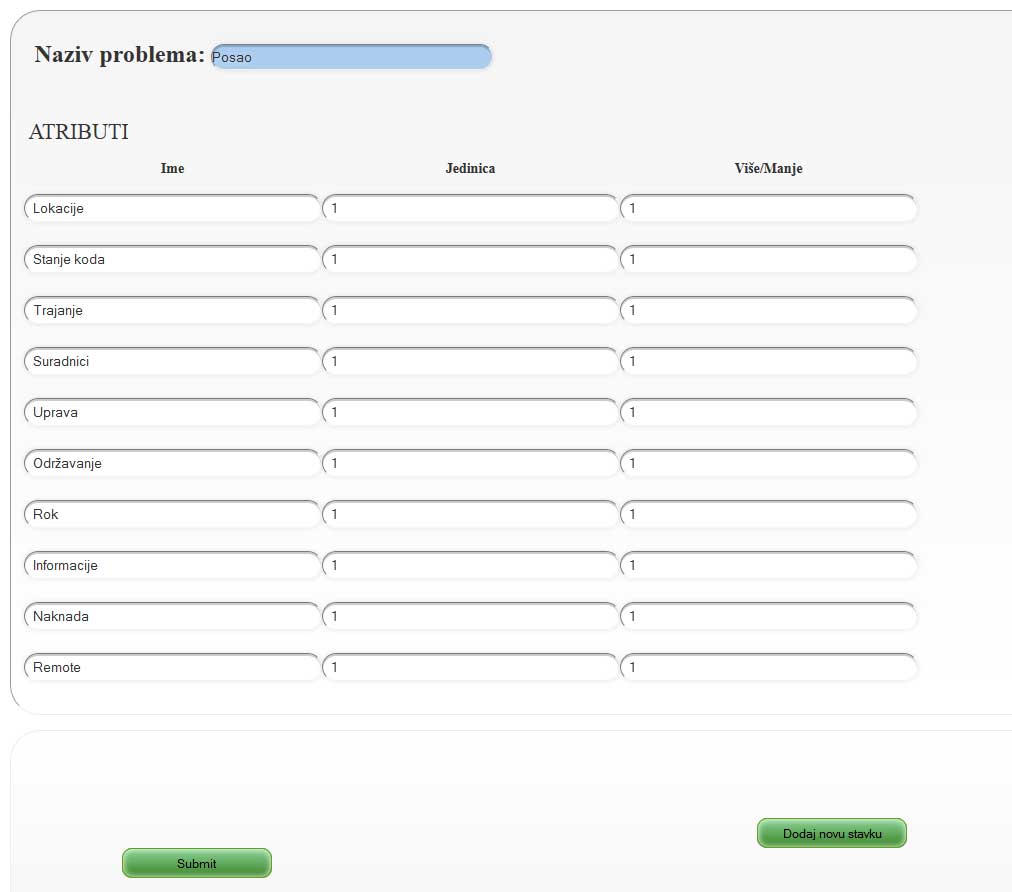
****

Slika 4. ES software – definiranje problema

Testni primjer kojega smo razradili u Poglavlju 2. sada ćemo definirati kroz slijed unosa kako bi ga mogli rješavati ES programom.

Definiramo kriterij, jedinicu mjere na koju se on odnosi, te način mjerenja: više je bolje, ili manje je bolje. To onda treba ponoviti za sve atribute.

Za navedeni primjer to izgleda ovako:

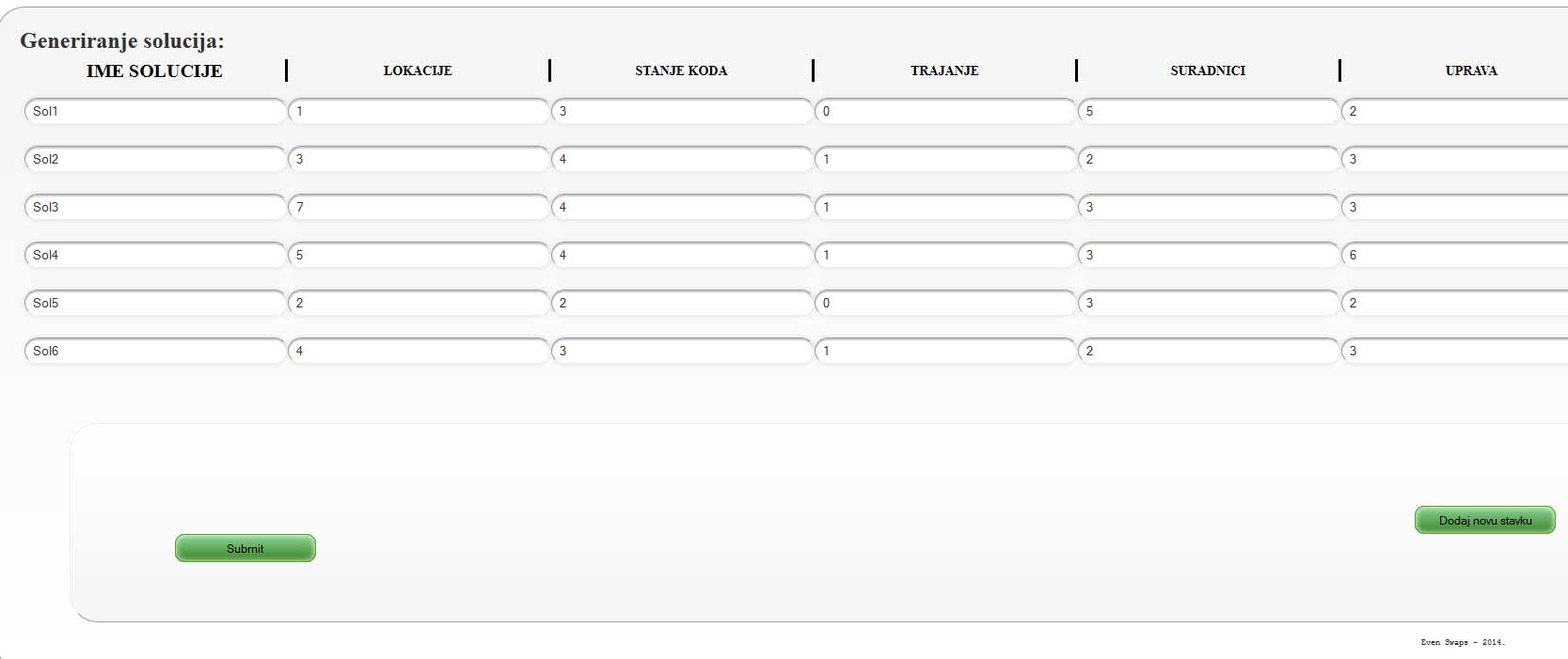
****

Slika 5. ES software – definirani atributi

Moguće je, naravno, Lokaciju mjeriti u stotinama kilometara, ili Nadoknadu u tisućama kuna, ili već kako donosilac odluke drži da mu je lakše.

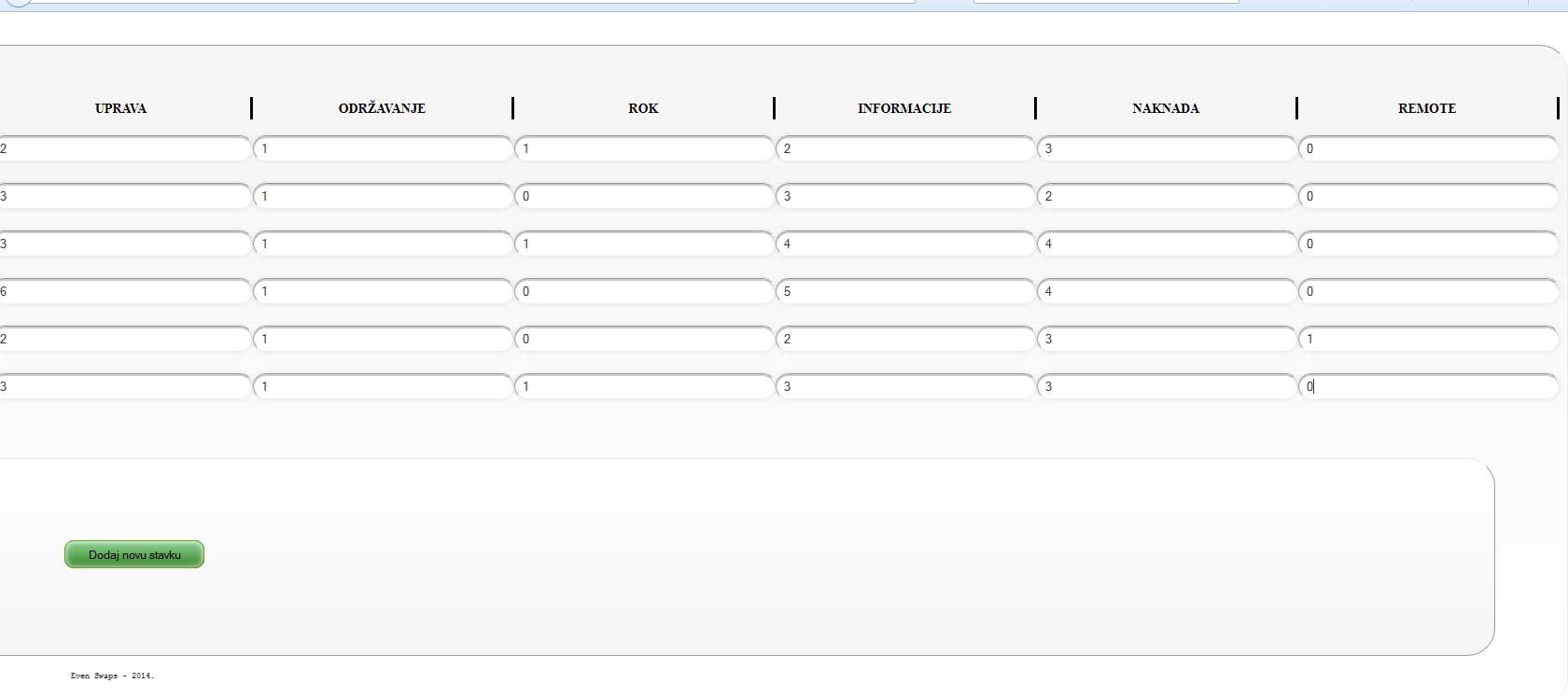
U ovom primjeru je odabran najjednostavniji pristup: svakako se međuodnosi definiraju kompenzacijskom tabelom, pa je jednostavnije razmišljati koliko na ukupnom rasponu vrijednosti Lokacije (1-9) vrijedi neka konkretna Naknada čiji je raspon također 1-9.

Nakon atributa generiramo solucije:



Slika 6. ES software – generirane solucije I

Program će posložiti u gornji red sve atribute koje smo definirali, te omogućiti generiranje proizvoljnog broja solucija koje ih koriste.



Slika 7. ES software – generirane solucije II

Kod generiranja solucija koristi se ili skala 1-9, ili „binarna“ 0-1. Za ovaj i ovakav slučaj to je dovoljan raspon. Pravi problem je međusobni odnos kod zamjena koji se definira kompenzacijskom tabelom:

****

Slika 8. ES software – kompenzacijska tabela

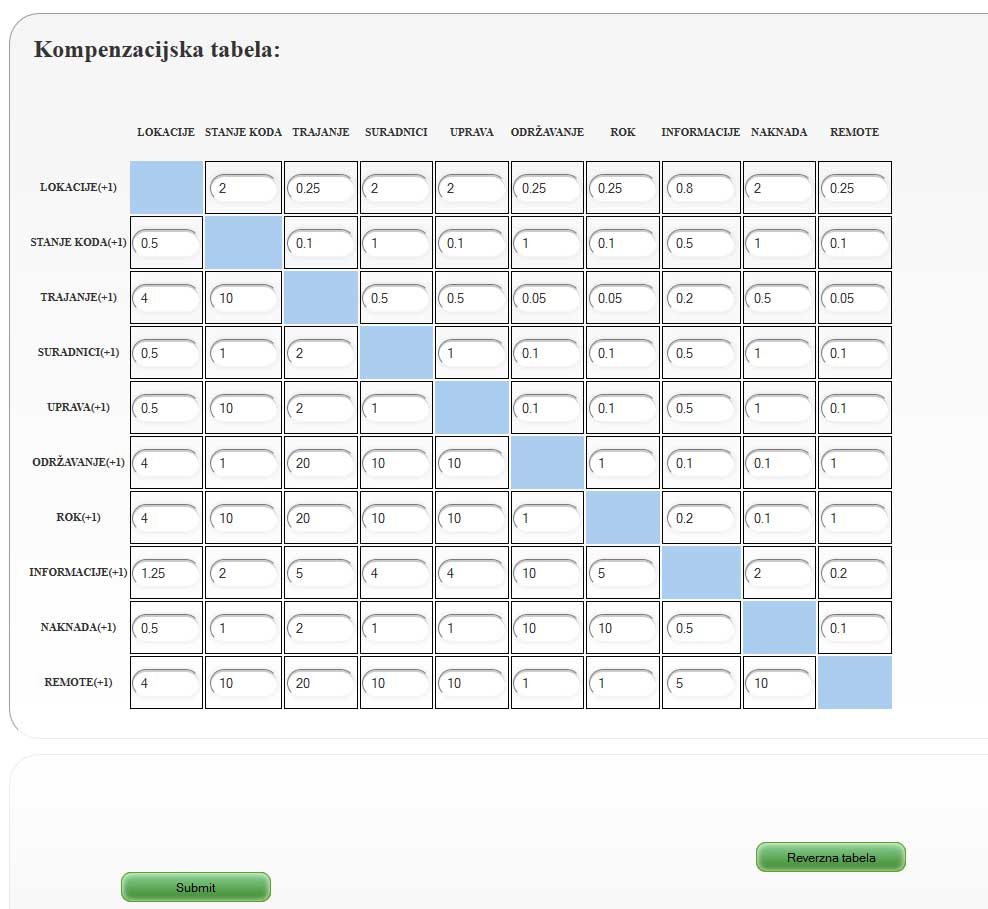
Moguća je u dvije varijante:

* Egzotičnoj Kompletnoj
* Uobičajenoj Reverznoj

**Kompletna tabela** se pokazuje i sa adresom korespondentnog polja tako da se korisnik ne može zabuniti kod definiranja odnosa atribut1 – atribut2 ili atribut2 - atribut1.

**Reverzna tabela** ne treba te pomoći, jer sve što se upiše sa gornje strane dijagonale automatski će se pojaviti u svome recipročnom obliku i na odgovarajućem mjestu sa donje strane dijagonale.

To izgleda ovako:



Slika 9. ES software – kompenzacijska tabela II

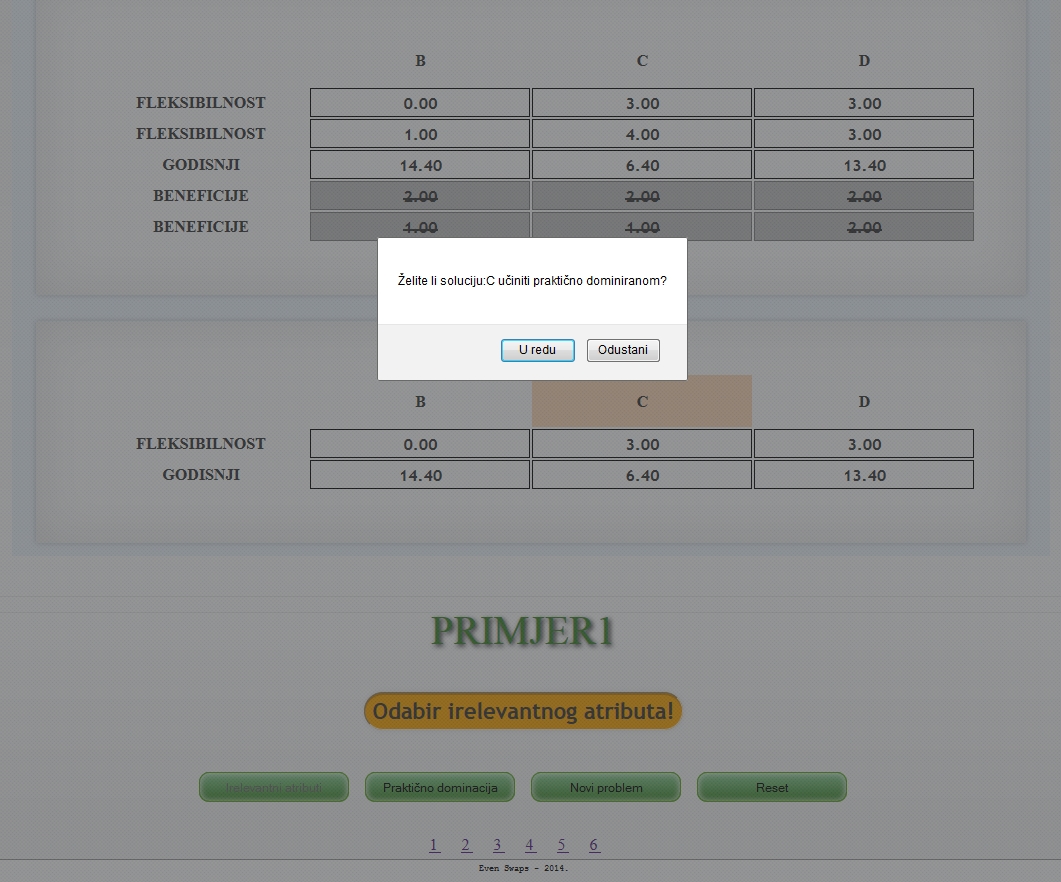
Dakle, STANJE KODA / LOKACIJA =2 isto kao i LOKACIJA / STANJE KODA=0,5.

Interesantan je „binarni“ REMOTE. To je vrlo važan kriterij budući se radi o poslu koji omogućuje rad na daljinu. Otuda i takvi odnosi kod kompenzacije: TRAJANJE/REMOTE=20. Priroda tog kriterija i jeste „binarna“ (Da/Ne), no tek se po kompenzacijskim kriterijima vidi do koje je mjere važan.

Kompenzacijski kriteriji mogu biti i negativni. Na taj način se improvizira odnos više/manje.

## 3.3 Aplikacija - sučelje

* **Reset** znači da se zamjene vraćaju u početno stanje, dakle na ovu situaciju u kojoj smo se našli kada smo došli na definirani problem.
* **Novi problem** znači da se odustaje od zamjena po odabranom problemu te se želi raditi na nekom drugom problemu ili definirati novi problem. To na odvodi na sam početak programa o kojemu je bilo riječi u prethodnom poglavlju.
* **Irelevantni atributi** se odnose na situaciju u kojoj se donositelj odluke odlučio jedan od atributa učiniti irelevantnim. Potrebno ga je označiti, te potom odabrati kompenzacijski atribut.
* **Praktična dominacija** obuhvaća obje vrste dominacije u ovome trenutku. Da bi donositelj odluke neku od solucija učinio dominiranom potrebno je kliknuti na nju. Ona postaje narančasta te gumb za dominaciju postaje upotrebljiv.
* **Navigacija kroz rješavanje problema:** potrebno je omogućiti korisniku da se može vratiti iz neke definirane situacije na točku s koje želi krenuti drugim putem. To izgleda ovako:



Slika 9. ES software – navigacija kroz korake zamjena

Ovdje treba primijetiti brojeve koji su se pojavili ispod menija. Klikom na zadnji broj dobiti ćemo samo zadnje stanje, umjesto dvije tabele jednu. Klikom na svaki prethodni broj dobiti ćemo prethodno stanje, dakle tako možemo ići do prvog +1 stanja. Na prvo stanje se vraćamo gumbom reset. Na taj način moguće je više puta probavati naći rješenje od neke upamćene točke.

## 3.4 Primjer Posao

Problem smo definirali u do sada opisanim koracima pa sada možemo probavati razmjene dok ne dođemo do zadovoljavajućeg rješenja.

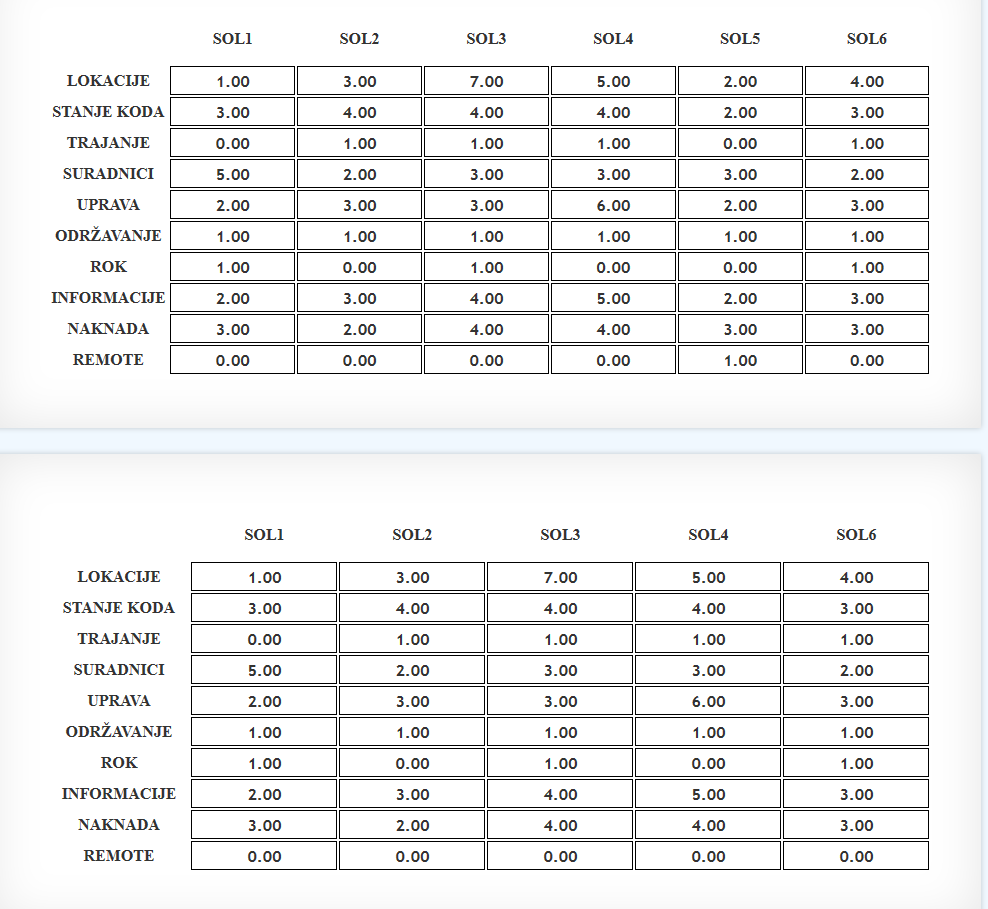


Slika 10. ES software – primjer Posao

Jedna od interesantnijih solucija bi mogla izgledati ovako: SOL5 je praktično dominirana od SOL4. Program nije dao takvu sugestiju iz dva razloga:

* Kod starta se samo pokaže stanje, a ne provjerava se ni startna irelevantnost atributa ili moguća dominacija solucija.
* SOL4 je bolja po svim kriterijima osim zadnjeg, REMOTE.

Kada se SOL5 ukloni iz projekta dobije se ovo:

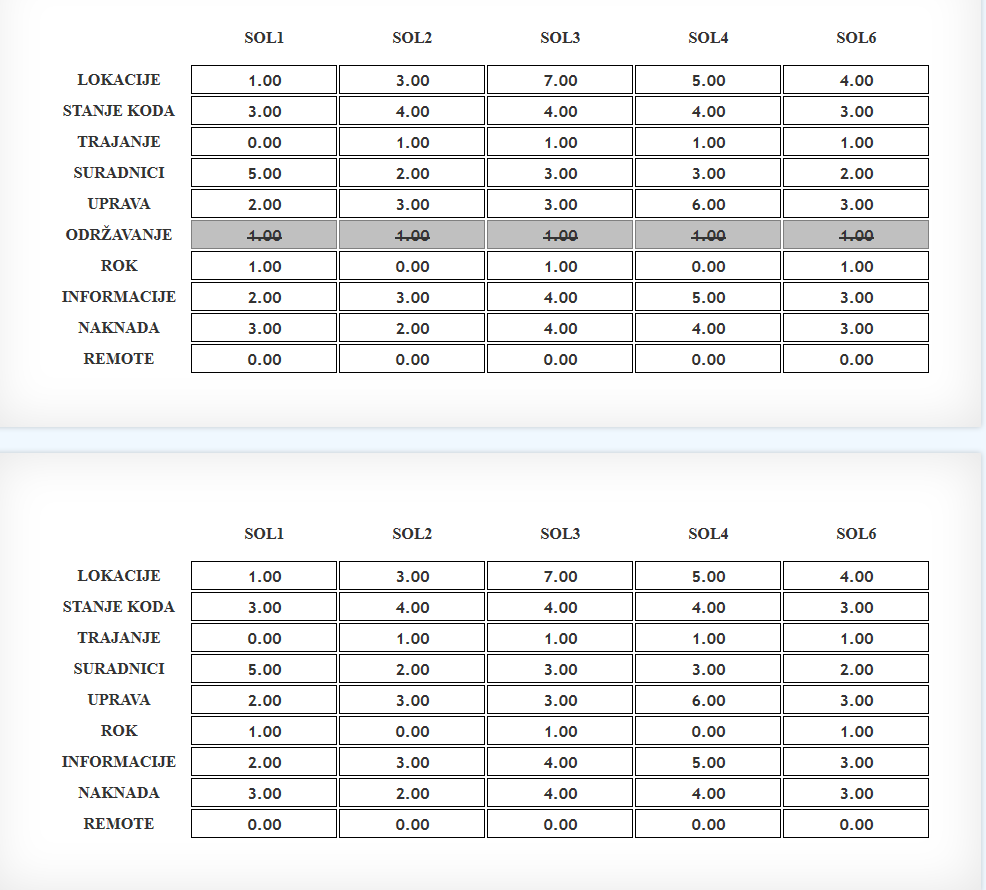


Slika 11. ES software – primjer Posao – zamjene I

Prvo početna situacija sa SOL5, pa zatim situacija bez nje.

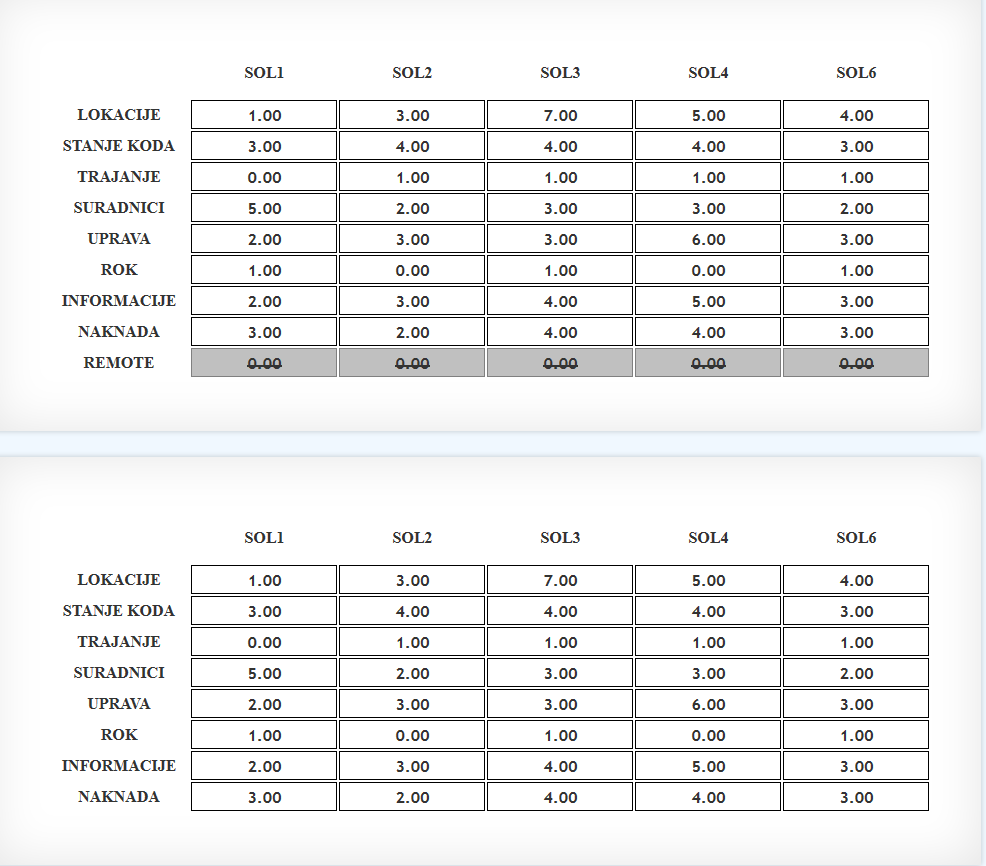
Na novonastaloj situaciji se vidi da su REMOTE i ODRŽAVANJE irelevantni atributi (REMOTE je to postao uklanjanjem SOL5 a ODRŽAVANJE je takav od početka).

I program je to uočio jer iza navedene dvije tabele slijede još četiri:



Slika 12. ES software – primjer Posao – zamjene II

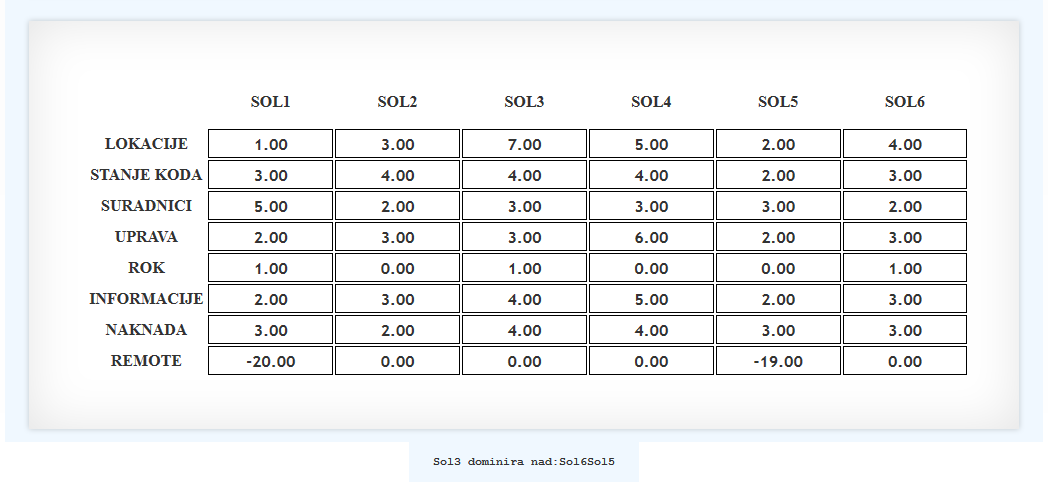
Te:



Slika 13. ES software – primjer Posao – zamjene III

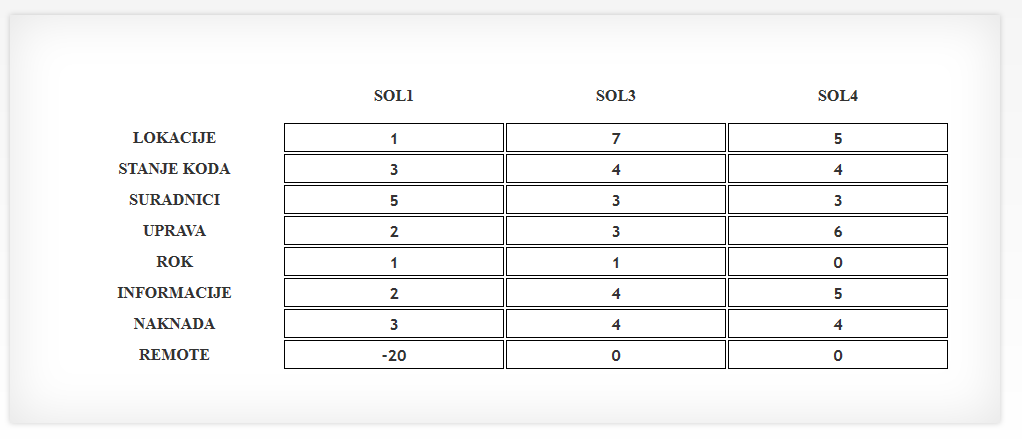
Sada se preko Reset vratimo na početak, pa idemo problem rješavati na neki drugi način.

Odaberimo Trajanje za irelevantni atribut te REMOTE za kompenzacijski. Program će i opet sam detektirati ORDŽAVANJE kao irelevantno, tako da se na kraju desi ovo:



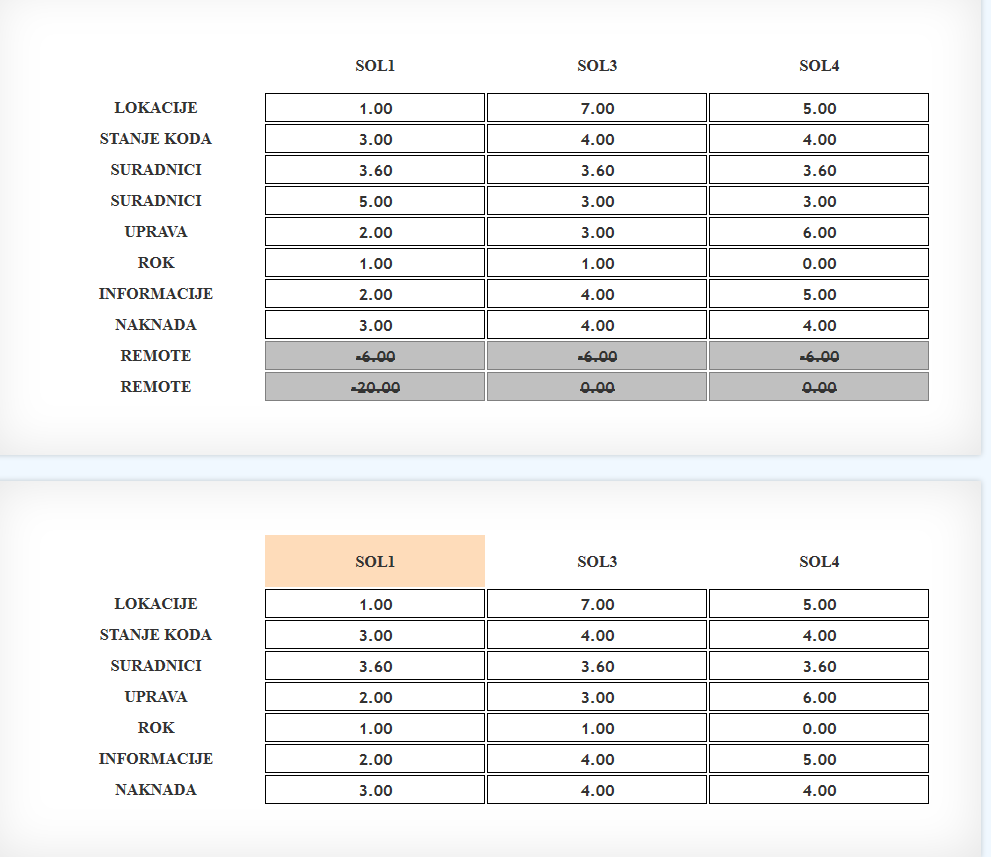
Slika 14. ES software – primjer Posao – zamjene IV

SOL3 dominira nad SOL5, SOL6 te SOL2. Za razliku od puta kojime smo krenuli maloprije imamo i relativno veliki REMOTE, -20.



Slika 15. ES software – primjer Posao – zamjene V

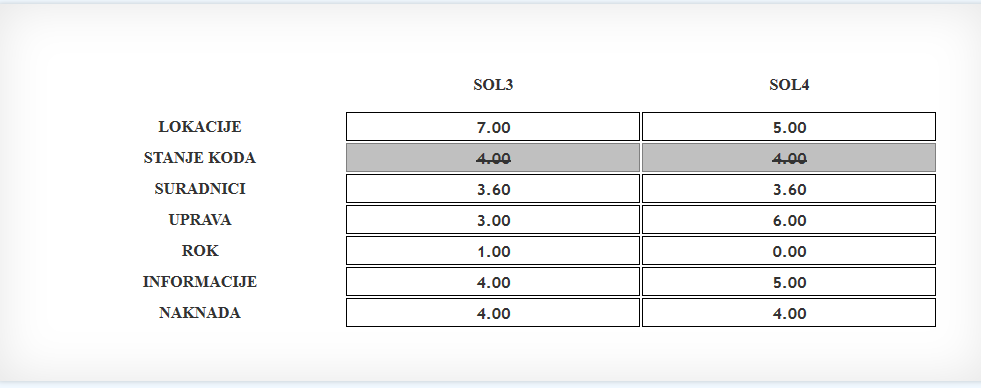
Irelevantni REMOTE, kompenzacijski SURADNICI:



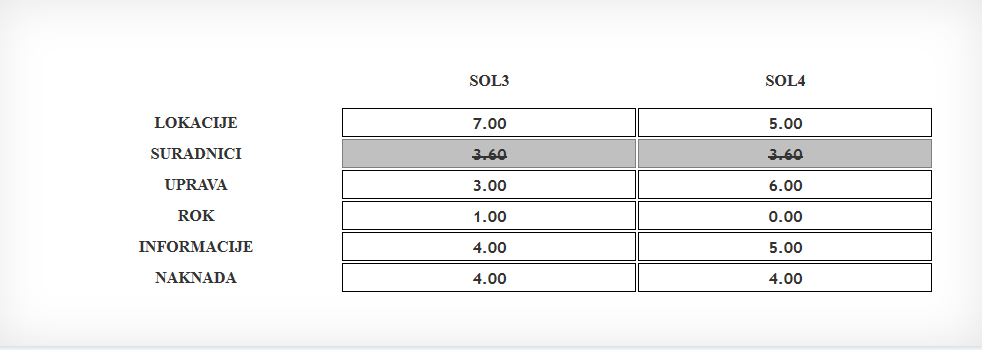
Slika 16. ES software – primjer Posao – zamjene VI

Kada se REMOTE svede na 6 (ceil(20/3)), pa se po kompenzacijskoj tabeli kompenziraju SURADNICI, ispasti će da je u svakoj soluciji vrijednost za SURADNIKA 3,6. To znači da su radi kompenzacije i SURADNICI postali irelevantni, te je SOL1 dominirana od SOL3.

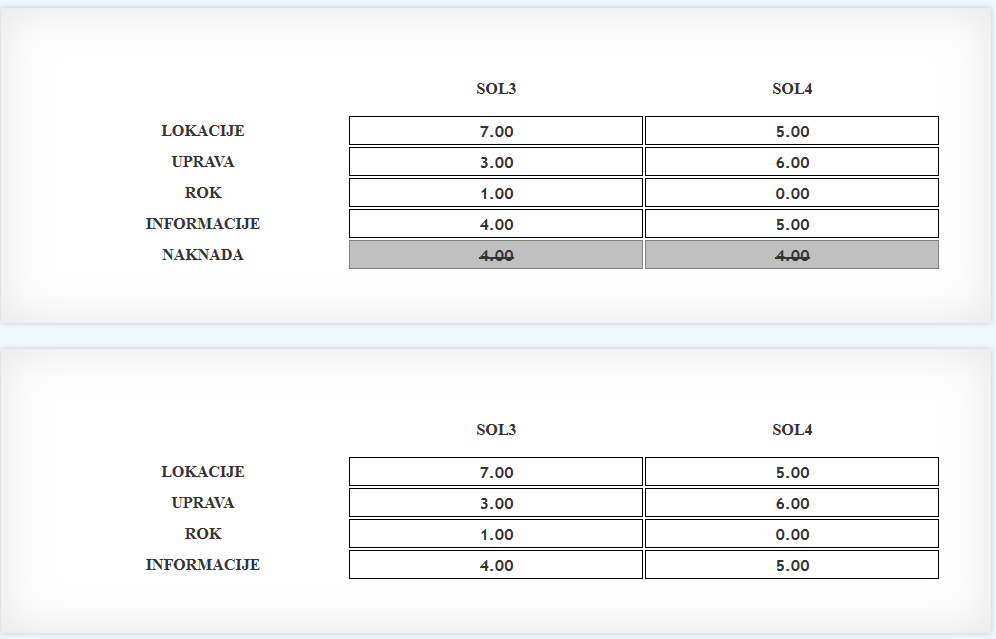
Micanje SOL1 će napraviti lančanu reakciju sa STANJEM KODA te NAKNADOM:



Slika 17. ES software – primjer Posao – zamjene VII



Slika 18. ES software – primjer Posao – zamjene VIII



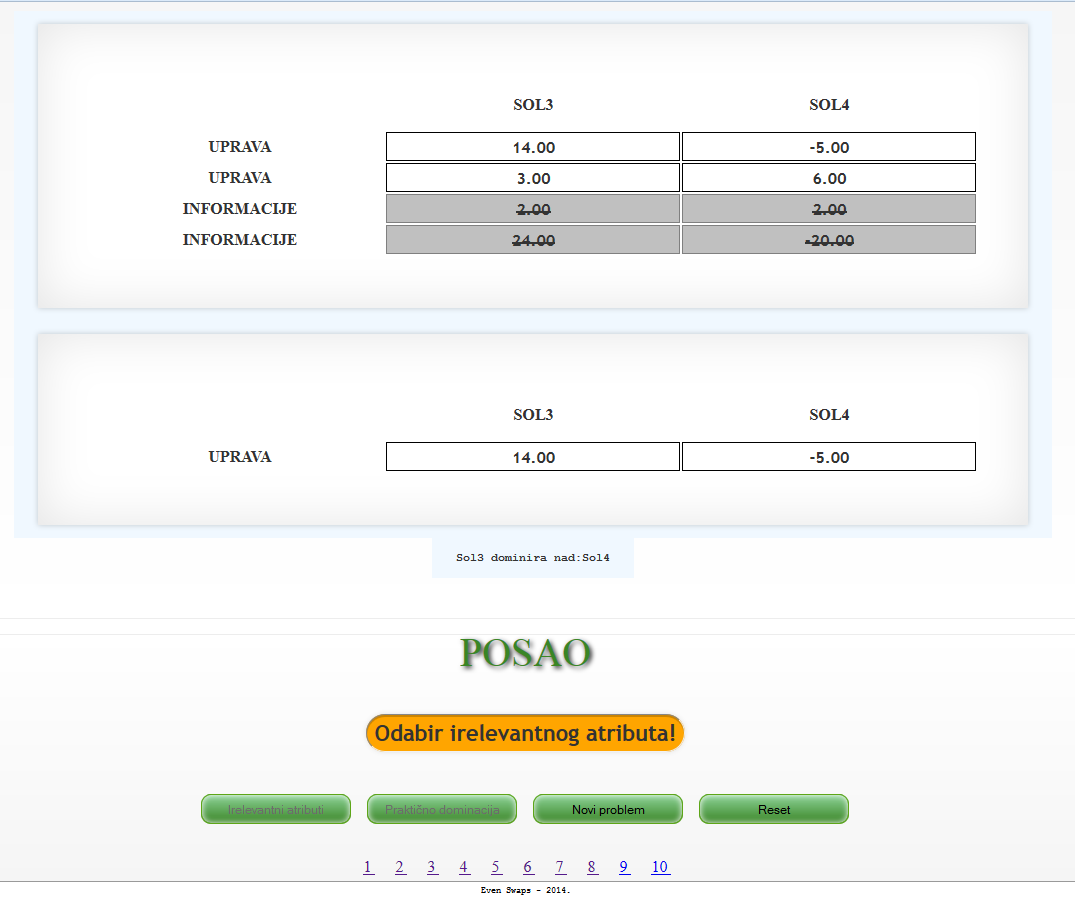
Slika 19. ES software – primjer Posao – zamjene IX

Nakon još jedne iteracije dolazimo do rješenja:



Slika 20. ES software – primjer Posao – zamjene X

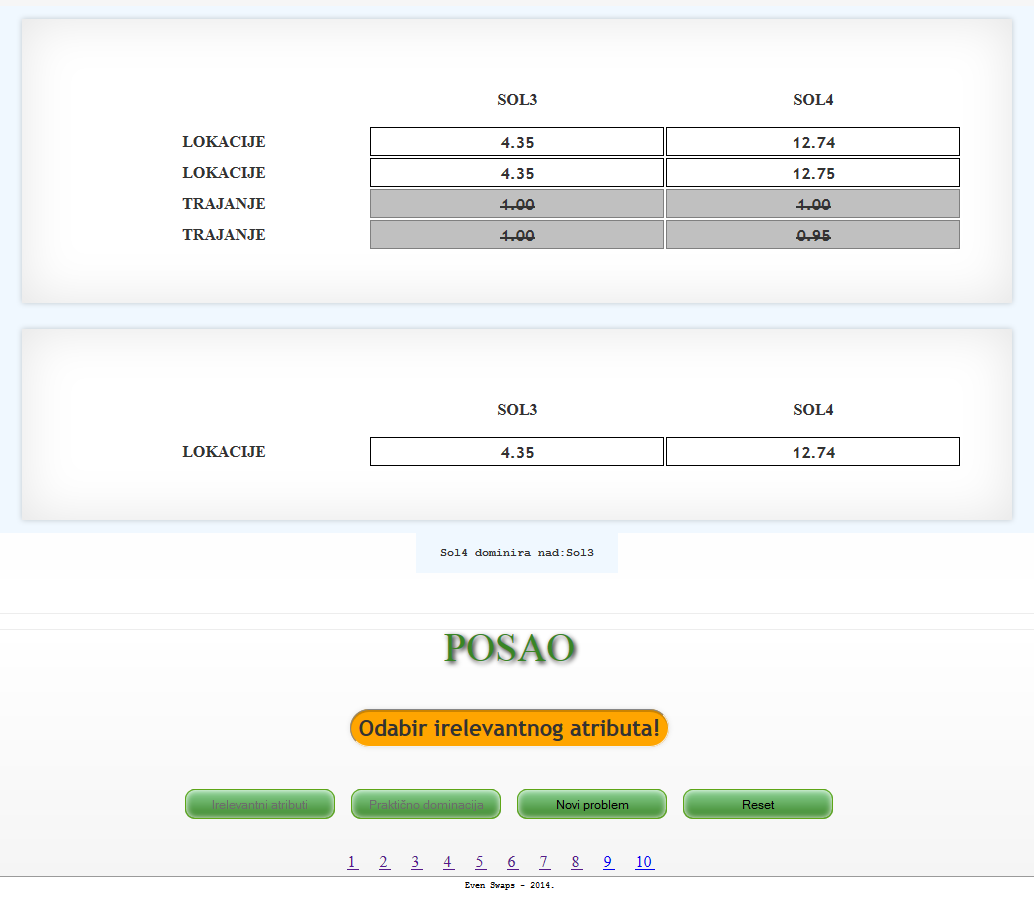
Da smo nastavili putom kojime smo krenuli u prvom pokušaju, došli bi do slične situacije: na kraju bi od atributa također ostali UPRAVA i INFORMACIJE, od solucija SOL3 i SOL4 no u obrnutom poretku: SOL3 bi dominirala SOL4, te bi nam do toga rezultata trebala dva koraka više.



Slika 21. ES software – primjer Posao – zamjene XI

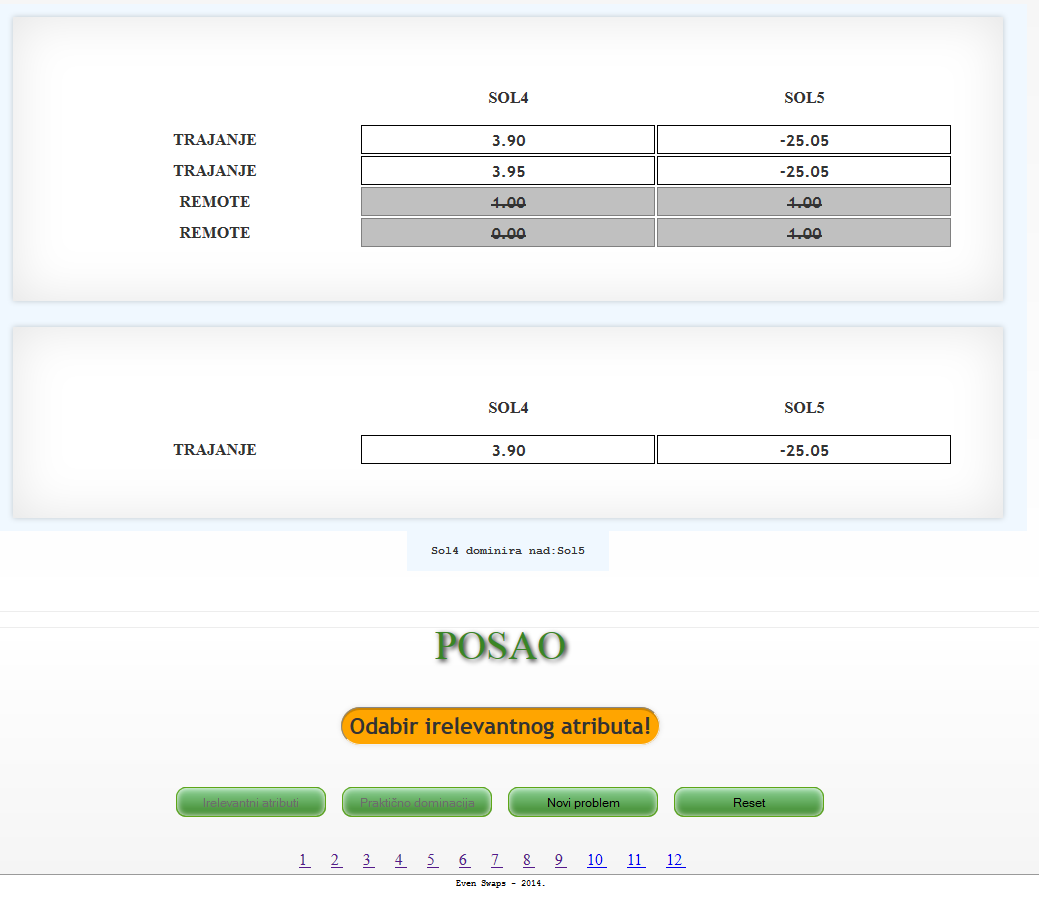
Probavano je i dalje, ovaj puta odmah uzevši za irelevantne atribute s kojima bih dolazio do kraja , UPRAVU i INFORMACIJE.

Na kraju je probano zamijeniti dva zadnja atributa, LOKACIJE i TRAJANJE, no rezultat je bio isti : SOL4 dominira SOL3



Slika 22. ES software – primjer Posao – zamjene XII

Situacija u kojoj REMOTE nije irelevantan skroz do zadnje iteracije eliminira SOL3 ali je SOL4 ipak dominantan preostalom SOL5:



Slika 23. ES software – primjer Posao – zamjene XIII

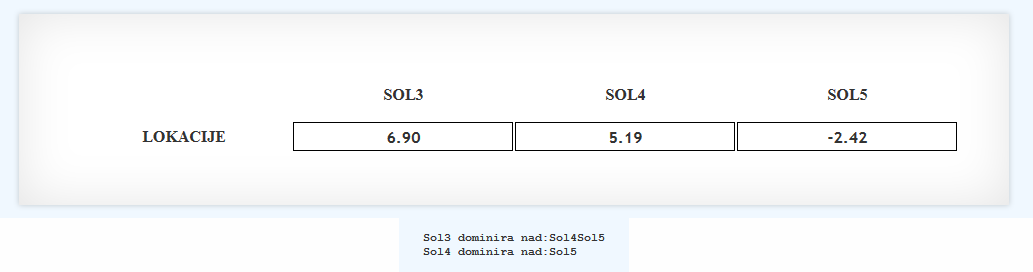
## 3.5 I Zaključak po primjeru

* Ako REMOTE postane irelevantan prije zadnje iteracije, dvije najbolje solucije su SOL4 i SOL3, dominira SOL4.
* Ako REMOTE postane irelevantan tek u zadnjoj iteraciji (jer se prije toga neće moći donijeti odluka) najbolje solucije su SOL4 i SOL5, dominira SOL4
* Ako se SOL5 ukloni po principu praktične dominacije (SOL4 je bolja po svim kriterijima osim REMOTE) najbolje solucije su opet SOL3 i SOL4 ali ovaj puta dominira SOL3

Uzmimo sada da se EVEN SWAPS koristi prvenstveno da bi se došlo do gornjih informacija. To znači da slijedi još jedna instanca odlučivanja koja bi moglo izgledati ovako:

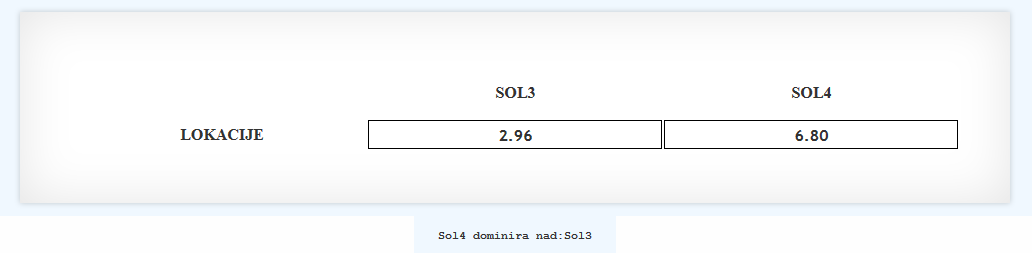
* Ili je najbolja solucija SOL4
* Ili treba u korelaciju sa SOL4 staviti važnost REMOTE pa se eventualno odlučiti za SOL3.

Ako odbacimo u startu SOL1, SOL2 i SOL6 te proglašavam atribute slijedno irelevantnima bez daljnjeg odbacivanja solucija, dobiti će se na kraju ovo:



Slika 24. ES software – primjer Posao – rješenje

Učinimo li isto to s time da u startu odbacim i SOL5, dobiti će se ovo:



Slika 25. ES software – primjer Posao – rješenje II

U konkretnom primjeru trebalo bi odvagnuti važnost kriterija REMOTE pa se možda ipak odlučiti za SOL4 i to zato jer je u pitanju daleki London. No, da je u pitanju nešto bliže, važnost kriterija REMOTE bi prevagnula. Iako, po svoj prilici, tada bi već i prikazani mehanizam imao jasniji odgovor budući bi vrijednost po kriteriju LOKACIJA porasla…

Uglavnom, zaključci nastali nakon višestrukog ponavljanja ostavljaju mogućnost daljnjeg zaključivanja u smislu utjecaja neke eventualne promjene ili dodatne rafinacije prema dobivenim rezultatima.

# EVEN SWAPS, Prolog i GA

Kako bi pratili različite pristupe, razvijene su dvije verzije programa kako je već rečeno u uvodu.

Dostupni su na adresama:

<http://www.w-app.net/evenswap/index.php>

http://evenswap.w-a-f.org

Na drugoj adresi se radi sa svođenjem irelevantnog atributa na srednju vrijednost kako je opisano u poglavlju tri.

Na prvoj se umjesto srednje vrijednosti svodi na nulu. Na taj je način vrijednost atributa kompenzacijska vrijednost.

## 4.1 Konzistencija Even swapa

Sljedeće zamjene su napravljene na obje verzije programa:

Location->Code

Code -> Duration

Duration-> Associates

Associates ->Management

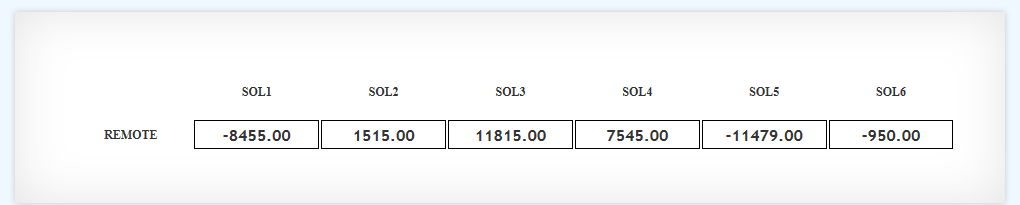
Management-> Deadline

Deadline-> Informations

Informations-> Wage

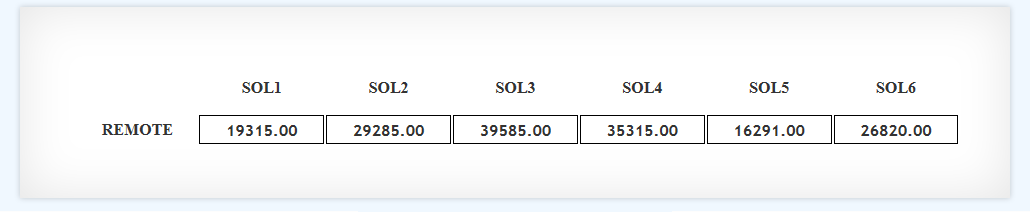
Wage -> Remote

**Srednja vrijednost:**



Slika 26. ES software – konzistencija Even swapa I

**Nula:**



Slika 27. ES software – konzistencija Even swapa II

**Sol3 je bolja od Sol4**. Odnosi između solucija su sačuvani:

11815-7545=4270

39585-35315=4270

Dakle, neovisno o broju na koji se svode irelevantni atributi razlika između solucija je ista.

Sada drugi redoslijed zamjena na istim programima:

Remote->Location

Location-> Wage

Wage-> Duration

Duration-> Management

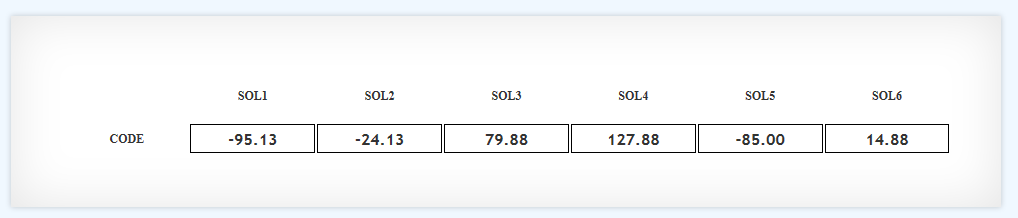
Management-> Deadline

Deadline-> Informations

Informations-> Associates

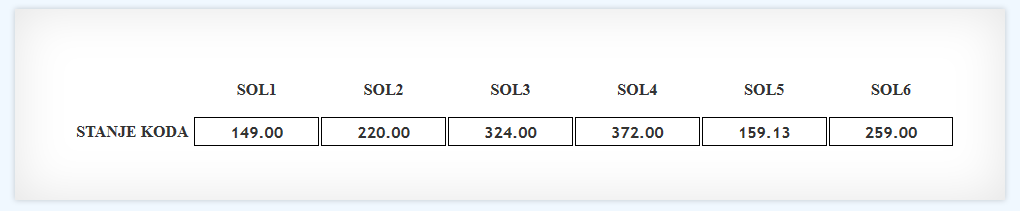
Associates->Code

**Srednja vrijednost:**



Slika 28. ES software – konzistencija Even swapa III

**Nula:**



Slika 29. ES software – konzistencija Even swapa III

I u ovom slučajurazlika između solucija je ista. No, sada je **Sol4 bolja od Sol3.**

Razlike nastaju radi različitih akumulacija prethodnih stanja koja su sadržana u novom stanju. Ako se svaki ciklus zamjene iskoristi za novu akumulaciju svih dotadašnjih zamjena govorimo o KASKADI. Ona se može predstaviti na ovaj način:

= f(n) + g(n)

F(n) se odnosi na množenje koeficijentom a g(n) na dodavanje vrijednosti atributa.

Koristeći taj obrazac dobije se veći rezultat od ostalih pristupa jer samo on ima množenje sa novim koeficijentom u svakoj iteraciji pa je – uz koeficijente koji su pozitivni i veći od jedan - najveća kaskada i najveći mogući rezultat.

Drugi pristup je ZVIJEZDA: akumulacija se odnosi samo na dodavanje novih vrijednost već prikupljenima. Kod generirane zvijezda se ne može rekonstrurati tijek zamjena.

Sve rečeno u poglavlju 3. o izboru nove zamjene uz ove činjenice izgleda bitno drugačije. Kako sada odabrati optimalan put?

Ako znamo što želimo možemo probati mijenjati prema željenom obrascu. I dok zvijezda ima samo n, kod kaskada je to bitno više:

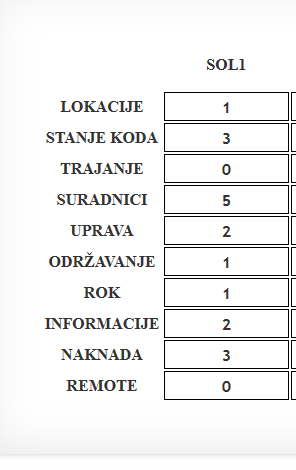
Prvi izbor može biti bilo kakav, dakle n(n-1) kombinacija jer je moguća zamjena bilo kojeg atributa sa bilo kojim drugim. No, već u sljedećoj iteraciji izbacujemo ta dva atributa koja se već nalaze u kaskadi pa ostaje na raspolaganju (n-2)(n-3)… Ukupni broj kaskada bi po tome bio n! što je za n!/n manje od ukupnog broja kombinacija even swapsa.

No, kako tražimo najveći mogući broj koji se kaskadom može dobiti, ukupni broj kombinacija je bitno manji. Tražimo kaskadu u kojoj će se množiti najveći koeficijenti. Kod even swapsa je barem polovica svih koeficijenata >= 1 pa se traže samo kombinacije koje sadrže samo njih. Na taj način se naprave veze između atributa koje određuju od kuda se kamo može ići, a to bitno reducira ukupan broj kombinacija. No, ne može se unaprijed znati kolika je ta redukcija.

Uglavnom, želimo li najveću soluciju, „ručne“ zamjene nisu opcija.

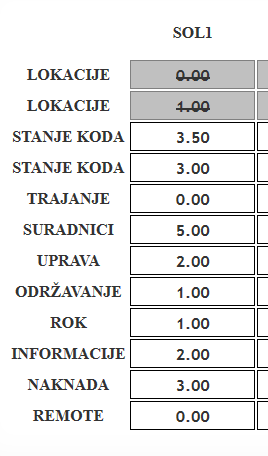
## 4.2 Zamjene sa svođenjem na nulu

Pogledajmo korak po korak što se događa u situaciji kada se irelevantni atribut svodi na nulu. Sol1 izgleda ovako:



Slika 30. ES software – zamjene sa svođenjem na nulu I

**Prva zamjena, Lokacije->Stanje koda:**



Slika 31. ES software – zamjene sa svođenjem na nulu II

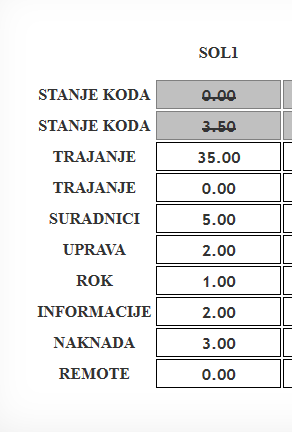
Svodeći irelevantni atribut na nulu vrijednost koja ulazi u kompenzaciju sa kompenzacijskim atributom je ista ona kakva je i zadana za danu soluciju. Iz kompenzacijske tabele se uzima koeficijent Lokacije/Stanje koda =0,5 pa se pola te kompenzacijske vrijednosti doda Stanju koda čija je nova vrijednost sada 3+0,5=3,5.

Recimo to ovako:

(L x kL) + Sk

(Lokacije x koeficijentLokacija) + Stanje koda

Hajdemo sada to tako dobiveno Stanje koda mijenjati dalje:



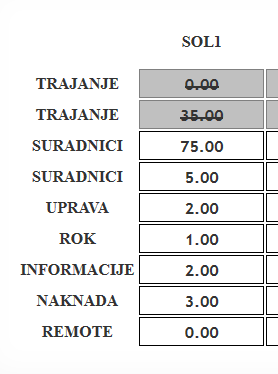
Slika 32. ES software – zamjene sa svođenjem na nulu III

Koeficijent Stanje koda/Trajanje je 10, pa imamo situaciju kao gore na slici.

((L x kL) + Sk) x kSk + T

((Lokacije x koeficijentLokacija) + Stanje koda) x koeficijent Stanja koda + Trajanje

Trajanje -> Suradnici:



Slika 33. ES software – zamjene sa svođenjem na nulu IV

(((L x kL) + Sk) x kSk + T) x kT + S

Ili

L x kL x kSk x kT + Sk x kSk x kT + T x kT + S

## 4.3 Kaskadne zamjene

Općenito kod kaskadnih zamjena:

(Ai ki)+ Aj čine jednu zamjenu,

((Ai ki)+ Aj) kii + Ajj dvije,

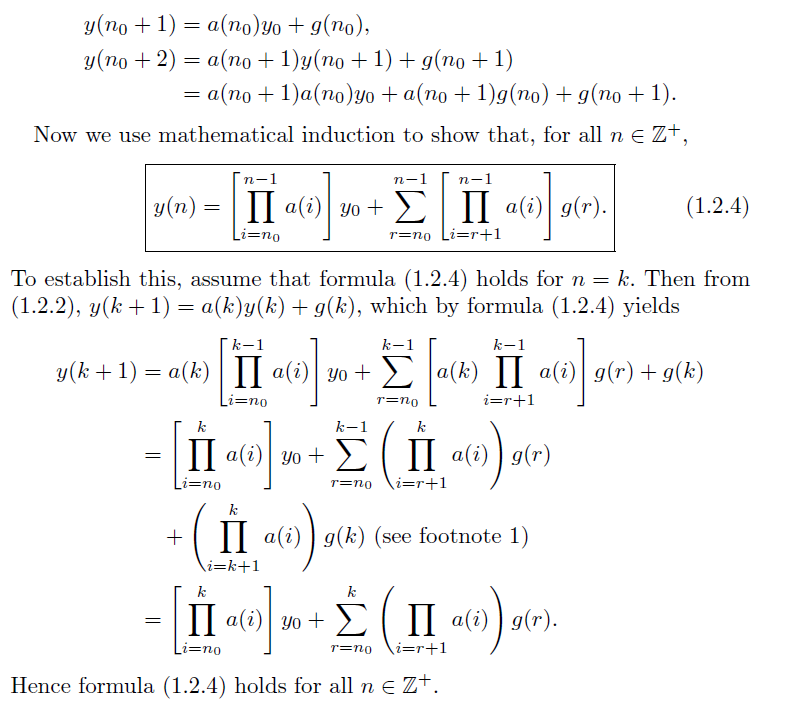
(((Ai ki)+ Aj) kii + Ajj) kiii + Ajjj tri, itd…

Ai ki kii kiii + Ajkii kiii + Ajj kiii + Ajjj …

Kaskadne zamjene:

Kada se zamjene obave na način da kompenzacijski atribut odmah u sljedećoj zamjeni postaje irelevantan, govorimo o kaskadi čija je ukupna vrijednost solucije dana gornjom formulom.

Kako je već rečeno, kaskada je nehomogena diferencna jednadžba prvog stupnja sa varijabilnim koeficijentima kod koje su svi koeficijenti >0. Prema Elaydi (Ref. 6.) izvod takve jednadžbe daje na prvi pogled nešto drugačiji rezultat:



Kao što se vidi na početku izvoda situacija potpuno odgovara kaskadi s time što funkcije (u prvoj iteraciji je to prvi atribut, nakon toga y sadrži dotadašnje stanje) i g(n) daju atribute a a(n) daje koeficijente. Taj prvi atribut možemo dobiti tako da u drugom množenju koeficijenti krenu od a ne od (i=r a ne i=r+1) pa se praktički prvi nađe u drugom. Upravo tako izgleda izvedena formula za kaskadu.

Nadalje, moguće je da uopće ne postoji na taj način definiran put kojime bi se mogli obići svi atributi. U tom slučaju se u postojeće staze dodaju i koeficijenti manji od jedan također poredani po svojoj veličini.

## 4.4 Opća zamjena

Opća zamjena je opći algoritam, pravilo koje uvijek vrijedi jer je direktan opis zamjene Even swapsa. Drugo je pitanje kakav je bio dotadašnji tijek zamjena te što je u kurentnoj zamjeni (na kojem je mjestu) prikupljena vrijednost a što novi atribut i pripadajući koeficijent.

Nova vrijednost kompenzacijskog atributa =

(zatečena vrijednost dosadašnjih zamjena ) X

koeficijent irelevantnog atributa +

vrijednost kompenzacijskog atributa

[AB]= (A x kAB)+ B

[BA]=( (A x kAB)+ B ) x kBA

= A + B x kBA

Označavanje osnovne zamjene kakvo ćemo koristiti kod razmatranja oblika. Ovdje je pokazana činjenica da osnovna zamjena pomnožena koeficijentom slobodni atribut -> vezani atribut mijenja stanje u suptono, pa slobodni atribut postaje vezan i obratno. Iskorišteno je svojstvo even swapsa da svaki koeficijent pomnožen sa suprotnim daje jedinicu, kAB x kBA =1.

U nastavku teksta koristiti će se isti način označavanja koeficijenata: kAB (koeficijent od A na B) znači da se radi o koeficijentu kojime se množe vrijednosti atributa A pri prijelazu na atribut B, A->B. kBA je suprotni put i recipročna vrijednost od kAB.

Ako je naveden samo jedan atribut, kA na primjer, onda se ne zna ili nije bitno (ili je iz situacije očito) za sam izvod kamo prelazi atribut A. Na tu se situaciju referira kao „koeficijent od A“.

## 4.5 Eliminacija koeficijenta manjeg od jedan

Kada bi svi koeficijenti bili veći ili barem jednaki jedan te svi atributi pozitivni do optimalne situacije bi ostao tek redoslijed.

Ako gornje zamjene promijenimo, pa nakon Lokacije->Stanje koda:

(L x kL) + Sk

Napravimo Trajanje->Suradnici

(T x kT) + S

Pa onda tako dobiveno Stanje koda -> u tako dobivene Suradnike

((L x kL) + Sk) x kSk + (T x kT) + S

L x kL x kSk + Sk x kSk + T x kT + S

Vidi se da to nije isto kao

(((L x kL) + Sk) x kSk + T) x kT + S

L x kL x kSk x kT + Sk x kSk x kT + T x kT + S

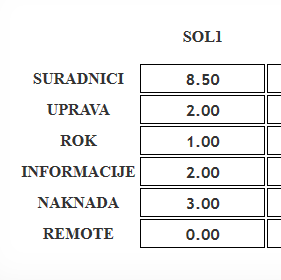
L – atribut Lokacija

Sk – atribut Stanja koda

T - atribut Trajanja

S – atribut Suradnika

U kaskadnoj varijanti su suradnici 75 a u ovoj drugoj 8,5.



Slika 34. ES software – zamjene sa svođenjem na nulu V

Te se dvije situacije razlikuju samo za množenje s kT-om, brojem većim od jedan (2) koji je dva sumanda u drugoj situaciji udvostručio. Dakle, ako postoji koeficijent manji od jedan, na ovaj način ga je moguće eliminirati iz dva množenja. No, to je izvedivo ako postoje 3 koeficijenta veća od jedan koje se može iskoristiti u tu svrhu. Pri tome nije svejedno ni kakve su vrijednosti atributa: ako su negativne radi zbrajanja ili manje od jedan radi množenja.

## 4.6 Od zamjena do optimizacije

Umnožak Ai ki (atribut x njegov koeficijent) postoji u svakoj situaciji zamjene. Ako je pozitivan i veći od jedan postaje kandidat za kaskadu. Ako nije onda treba utvrditi je li to radi koeficijenta ili radi atributa.

Potrebno je dakle napraviti rangiranje Ai ki te posebno rangiranje ki .

Za svaki Ai ki >1 gdje je i ki >1 najefikasnije je rješenje sa najviše množenja, dakle kaskada.

*(Postavlja se pitanje što ako su to vrijednosti neznatno veće od jedan i ima ih puno u odnosu na ukupan broj atributa: nije li tada bolje izbjeći svaki četvrti ki <1 umjesto neznatnog povećanja kaskade?)*

Za ostale ostaje izbjegavanje najnižih koeficijenata.

Sve to treba jasno odvojiti jer po općoj zamjeni zadnji u nizu se zbraja. To je važno jer nema smisla kaskadom generirati neki veliki broj ako će u nekom množenju biti pomnožen malim koeficijentom.

**Algoritam u općem slučaju bi dakle išao ovako:**

* rangirati Ai ki
* rangirati ki
* odrediti kandidate za kaskadu
* odrediti zadnjeg za kaskadu
* optimizirati kaskadu
* optimizirati ostatak
* zadnji od kaskade je kompenzacijski za ostatak

## 4.7 Ravnomjerne zamjene

Opći slučaj podrazumijeva mogućnost različitih koeficijenata za suprotne operacije. Lako je zamisliti takve situacije: nositi 10 Kg uzbrdo za 10 Kn/Km može biti prihvatljivo, ali za 10 lipa nizbrdo vjerojatno nije.

Zašto?

Zato jer je u pitanju složeni pojam: nošenje uzbrdo i nošenje nizbrdo se mogu usporediti po nekoj funkciji koja nije recipročna. Naime, pojam uzbrdo napravi reciprocitet prema pojmu nizbrdo, ali se to ne reflektira na masu pa je tako 10 Kg uzbrdo isto tako 10 Kg nizbrdo, a ne 100 grama kako bi to u ovoj situaciji ispalo. Nesumnjivo je lakše nositi teret nizbrdo, ali sasvim sigurno taj omjer za 10 Kg nije 100 puta koliko bi u ovom primjeru bio.

Sučelje razvijenog programa dopušta tu mogućnost: može se izabrati kakva kompenzacijska tabela se želi. U općem slučaju neće program sam generirati recipročne vrijednosti na odgovarajućem mjestu već to korisnik treba uraditi sam. Sučelje dakle radi, ali ta mogućnost još nije implementirana na serveru.

Već i po imenu Even Swaps se ne bi trebao odnositi na taj opći slučaj. Ravnomjerne zamjene podrazumijevaju recipročnu vrijednost za „obrnuti smjer“.

To znači da kompenzacijska tabela sadrži polovicu koeficijenata većih ili jednakih jedan, a to do sada opisane situacije bitno mijenja. Prvo pitanje koje se postavlja je mogućnost kaskade koja bi se dala zaključiti po kompenzacijskoj tablici.

Zamislimo atribut koji ima sve „pozitivne“ veze, dakle

A->B daje uvijek koeficijent veći od jedan. To znači da ne postoji „pozitivna“ relacija X->A jer su svi koeficijenti prema tom atributu manji od jedan. Od njega lako možemo otići ali nemamo kako k njemu doći.

On onda može biti prvi u nizu, a „kolega“ kome se može doći a ne može otići zadnji. Postoji li više takvih slučajeva kaskada se ne može izvesti.

Postoji li barem jedan A->B gdje je koeficijent manji od jedan, onda može postojati način i da se taj atribut uključi u kaskadu. Može, ali i ne mora jer se istri problem može javiti i sa strane tog atributa.

Takve se atribute može skloniti, a prikladna zamjena problem učiniti jednostavnijim.

## 4.8 Kompenzacijska tabela - prologova baza podataka

Prema (Ref. 7.) Prolog je: „*Programski jezik za logičko programiranje, koji je 1970. razvio francuski informatičar Alain Colmerauer na Sveučilištu u Marseilleu. Program napisan tim jezikom sastoji se od činjenica i pravila koji se odnose na neko područje primjene. Za razliku od tradicionalnih programskih jezika, koji prvenstveno obrađuju brojčane podatke, PROLOG obrađuje simbole koji su među sobom povezani pravilima simboličke logike. Ugrađenim postupcima logičkoga pretraživanja program oponaša ljudsko zaključivanje, pa je PROLOG 1980-ih bio u Europi i Japanu jedan od glavnih jezika u području istraživanja*[*umjetne inteligencije*](http://www.enciklopedija.hr/natuknica.aspx?ID=63150).“

Prolog u radu koristi bazu činjenica. One mogu biti jednostavan opis situacije (muško(Pero):-true.) no ovdje je prvenstveno interesantna činjenica veze između atributa (veza(A,B,4):-true. To bi značilo kAB=4). Ako u relaciju veza(\_,\_,\_) smjestimo sve veze između atributa te njihove pripadajuće koeficijente dobili smo prologovu bazu činjenica.

Uzmemo li sve trojke Irelevantni – Kompenzacijski – Koeficijent takve da je koeficijent >=1 i sortiramo to silazno dobije se listing iz Priloga 1. Ukupni broj unosa u kompenzacijsku tabelu je n\*n-n a podijeljen je na tri dijela: dio s koeficijentima većim od jedan, manjim od jedan te jednakim jedan. Prvi i treći se mogu iskoristiti za konstrukciju najveće kaskade.

Sada od te tabele možemo napraviti prologovu bazu: tabela sadržava „pozitivne “ veze Irelevantni->Kompenzacijski čak i kada je koeficijent jedinica: ne može li se drugačije spojiti dvije točke, dobra je i jedinica. Bitno je da nije manji od jedan.

Između ostaloga prolog se koristi i za prolaz kroz grafove i stabla. A ovako konstruirana prologova baza činjenica predstavlja težinski graf koji na žalost može imati cikluse.

Prologom se mogu generirati sekvence između neka dva čvora. Mogu se generirati dijelovi kromosoma za genetski algoritam pogotovo ako nisu veliki.

U Prilogu 5 se nalazi prolog programski kod kojime se mogu dobiti sve moguće kombinacije obilaska tako definiranog stabla.

U Prilogu 6 se nalazi prologova baza činjenica dobivena iz tabele sortiranih koeficijenata.

U Prilogu 7 se nalazi listing programa, svi putovi koje je prolog pronašao.

Nisu svi ti putovi kompletni, zapravo velika većina ne sadrži kompletan prolaz kroz graf. A i oni koji jesu dovršeni su tek jedna od mogućnosti među kojima se traži ona sa najvećim umnoškom.

## 4.9 Genetski algoritam - generiranje kaskade

Prema (Ref. 8.) Genetski algoritam je:“ *Genetski ili genetički algoritam (GA)  je heuristička metoda optimiranja koja imitira prirodni evolucijski proces. Evolucija je robustan proces pretraživanja prostora rješenja. Živa bića se tijekom evolucije prilagođavaju uvjetima u prirodi, tj. životnoj okolini. Analogija evolucije kao prirodnog procesa i genetskog algoritma kao metode optimiranja, očituje se u procesu selekcije i genetskim operatorima. Mehanizam odabira nad nekom vrstom živih bića u evolucijskom procesu čine okolina i uvjeti u prirodi. U genetskim algoritmima ključ selekcije je funkcija cilja, koja na odgovarajući način predstavlja problem koji se rješava. Slično kao što su okolina i uvjeti u prirodi ključ selekcije nad nekom vrstom živih bića, tako je i funkcija cilja ključ selekcije nad populacijom rješenja u genetskom algoritmu. Naime, u prirodi jedinka koja je najbolje prilagođena uvjetima i okolini u kojoj živi ima najveću vjerojatnost preživljavanja i parenja, a time i prenošenja svojega genetskog materijala na svoje potomke. Za genetski algoritam jedno rješenje je jedna jedinka. Selekcijom se odabiru dobre jedinke koje se prenose u slijedeću populaciju, a manipulacijom genetskog materijala stvaraju se nove jedinke. Takav ciklus selekcije, reprodukcije i manipulacije genetskim materijalom jedinki ponavlja se sve dok nije zadovoljen uvjet zaustavljanja evolucijskog procesa. Konačan rezultat je populacija jedinki (potencijalnih rješenja). Najbolja jedinka u zadnjoj iteraciji predstavlja rješenje optimiranja*.“

Prema (Ref. 9.):“ *Genetički* [*algoritam*](https://hr.wikipedia.org/wiki/Algoritam) *oponaša prirodnu* [*evoluciju*](https://hr.wikipedia.org/wiki/Evolucija_%28biologija%29)*, tj. selekciju.*

*Stvori se inicijalna* [*populacija*](https://hr.wikipedia.org/wiki/Populacija)*, svaka jedinka s vlastitim* [*genomom*](https://hr.wikipedia.org/wiki/Genom)*. Tada se svaka jedinka ocjenjuje posebnom funkcijom koja se zove* [*funkcija cilja*](https://hr.wikipedia.org/w/index.php?title=Funkcija_cilja&action=edit&redlink=1) *(*[*engl*](https://hr.wikipedia.org/wiki/Engleski_jezik)*.* [*fitness function*](https://hr.wikipedia.org/w/index.php?title=Fitness_function&action=edit&redlink=1)*). Funkcija vraća vrijednost koja nam govori koliko je* [*jedinka*](https://hr.wikipedia.org/wiki/Jedinka) *"sposobna". Nakon što se svaka jedinka procijeni, poredaju se po vrijednostima funkcije cilja i iz populacije se izbacuju oni s najnižom vrijednosti. Tada se* [*genomi*](https://hr.wikipedia.org/wiki/Genom) *različitih jedinki rekombiniraju, događaju se* [*mutacije*](https://hr.wikipedia.org/wiki/Mutacija) *i tako se stvori sljedeća generacija. Tada se ponovno jedinke ocjenjuju funkcijom cilja to se ponavlja dok se ne dobije optimalno rješenje.“*

U konkretnom slučaju generiranje kromosoma je komplicirano

radi unutrašnjeg ustroja kaskade. Funkcija cilja mora provjeriti i raspored i umnožak što je dodatni problem.

Kod velikog broj atributa može se umjesto sa gen-atribut

kromosomom raditi sa parom Irelevantni-Kompenzacijski. Tabela sortiranih koeficijenata omogućuje razne kombinacije: niz od nekoliko Irelevantni-Kompenzacijski parova u mini-kaskadi ili paketu izbjegavanja malih koeficijenata ako se već ne bi mogao zatvoriti drugačije put.

Opisati će se algoritam primjenjiv za opću Swaps metodu, dakle gdje recipročna vrijednost suprotnih atributa nije obavezna. No, primijeniti će se na opisani primjer čija kompenzacijska tabela zadovoljava uvjet ravnomjernih zamjena.

U Prilogu 2 se nalaze sortirane vrijednosti atributa po solucijama. Iz njih je vidljivo koji atributi imaju najveći iznos. Njih je potrebno postaviti na sam početak kromosoma jer ih množi najviše koeficijenata.

S druge strane najveći koeficijent treba biti postavljen na drugi kraj kromosoma kako bi množio što više atributa. Kako se zadnja vrijednost samo zbraja, bez množenja, ako je potrebno radi zatvaranja kromosoma, najbolje mjesto za atribut kojega nema sa „prave“ strane tabele koeficijenata je upravo na kraju kromosoma: na taj se način dobije jedno „mjesto“ za gen boljih svojstava. Potrebno je pomaknuti na lijevo umnoške koeficijenata i atributa što u startu smanjuje rezultat, ali možda se ne može drugačije.

Dakle, veličina atributa je poredana opadajuće sa lijeve strane na desnu, dok je veličina koeficijenata poredana obratno, sa desne strane na lijevu.

Osim toga geni ne mogu biti poredani bilo kako, već se mora slijediti logika puta ((Irelevantni-Kompenzacisjki)-Irelevantni)-Kompenzacisjki… definirana tabelom sortiranih koeficijenata.

Osim ova dva sortiranja i grafa, treba paziti i na veličinu „reza“: kako je potrebno u startu definirati dvije krajnje točke, postavlja se pitanje koliko treba biti dugačka svaka od njih.

Algoritam implementiran na <http://www.w-app.net/evenswap/index.php> radi na sljedeći način:

1. Prvo se odvoji dio tabele sa koeficijentima i izabere se jedan od njih.
2. Ako generiramo konstrukciju, onda se ide po tabeli, od vrha prema dolje jer nam trebaju najveći koeficijenti, i traži se po uvjetima I->R tko najbliži vrhu zadovoljava taj uvjet. Nađe li se odgovarajuće atribute spoji ih se sa onim generiranim u prvom koraku.
3. Ako se u cijeloj tabeli takvo što ne nađe, onda se traži atribut koji bi prema uvjetu I->R odgovarao atributu izabranom u prvom koraku. Nađe li ga se istom tehnikom (dakle samo graf, ne vodi se računa o veličini koeficijenata) onda se traži i treći.
4. Ako tražimo koeficijente s GA, onda se prvi generira random a drugi traži po I->R uvjetu u određenom području tabele koeficijenata, dakle vodi se računa i o iznosu koeficijenata. Ne nađemo li ga ponavljamo prvi korak ali sa većim područjem pretraživanja tabele koeficijenata. Treći puta je područje pretraživanja cijela tabela koeficijenata. Prve 4 do sada opisane točke su dio algoritma getLast.
5. Slijedeći korak je getFirst. Prije njega je potrebno od sortirane tabele atributa oduzeti ono što je odabrao getLast. getFirst odabire početak kromosoma iz druge tabele, sortiranih vrijednosti atributa jer početak mora imati što veće atribute. Traži 2 atributa i to na random način (GA) ili ih konstruira na temelju I->R uvjeta i sortirane tabele atributa.
6. Sada treba oduzeti i atribute odabrane sa getFirst pa se ostatak zove getMiddle. On se prvo proba naivno: I->R uvjetima spojiti kraj početka i početak kraja nadajući se da je sve između posloženo u skladu s njima.

Kada to ne prođe onda random odabire prvog u middleu tako da odgovara I->R uvjetima sa zadnjim iz getFirst. I onda tako po redu do zadnjeg middlea koji mora odgovarati prema prvom iz getLast.

Kako rezultat toga izgleda na konkretnom kromosomu može se vidjeti u Prilogu 4.

1. Mutiranje: random se izaberu dva gena pa se provjerava može li se zamjena uklopiti u I->R pravila.
2. Proces počinje odabirom načina za getLast: 60% se generira konstruiranjem i 40% sa GA.
3. Zatim odabirom veličine dijela kromosoma kojega radi getLast: on uvijek napravi 3 gena, no 90% je vjerojatnost da će se treći, pomoćni, otkinuti te da će ostati koeficijentno (potencijalno) najjača varzija.
4. Nakon što se middle generira spajaju se tri dijela u cjelinu i idu na mutiranje.
5. Svakoj od tih akcija se daje ime te se zapisuje zajedno sa kromosomom. Solution origin govori kako je koji dio kreiran te u kojoj generaciji se to desilo.

## 4.10 **Genetski algoritam - rezultati**

Nakon eksperimentiranja sa brojem generacija kao i sa random veličinama kod generiranja dijelova, ispalo je da bi ovo sada na mreži moglo biti optimalno stanje. Poveća li se broj generacija za 6 puta u prvih dvadeset će se nalaziti 6-7 različitih kromosoma. Na ovoj razini broja generacija program radi bitno brže, no ne daje svaki puta iste rezultate. Treba ga više puta pokrenuti da bi bilo jasno koja solucija je najbolja.

PRVIH DVADESET od ukupno 3827:  
  
  
Solution name:**Sol1**  
Solution path:Suradnici - Stanje koda - Uprava - Lokacije - Trajanje - Održavanje - Informacije - Remote - Rok - Naknada  
Solution origin:Last:create-2. First:create. 297 main generation.  
Solution score:**6600613**

Solution origin kaže da je prvi u nizu generiran, dakle nije upotrebljavan GA. Zadnja dva također su kreirana iz tabele. Sve ostalo je uradio GA. Ova verzija se desila u 297 glavnoj generaciji.

6x manji broj generacija:

PRVIH DVADESET od ukupno 635:  
  
  
Solution name:**Sol1**  
Solution path:Suradnici - Stanje koda - Uprava - Lokacije - Trajanje - Održavanje - Informacije - Remote - Rok - Naknada  
Solution origin:Last:create-2. First:create. 131 main generation.  
Solution score:**6600613**

I kod manjeg generacija rezultat je isti, no nije broj utvrđivanja pojedine solucije na njenom poretku. To otežava rangiranje pa je potrebno cijeli proces više puta ponoviti.

## 4.11 II Zaključak po primjeru

Rezultati su iznenađujući. Nikada se nije korak-po-korak uspjelo dobiti za najbolju soluciju ništa osim Sol3 ili Sol4. One imaju najveće atribute i očito se čovjek kod izbora zamjena vodi upravo brojevima koje vidi. Ništa od toga se ne spominje u savjetima „umjetnosti zamjena“ opisanima u 2. poglavlju.

Također je korak-po-korak zamjenama teško dobiti ovolike brojeve. Obzirom na ukupan broj mogućnosti nije ni čudo da se slučajno ne može tek tako naići na potpuni prolaz kroz graf kojih ima tek nekoliko.

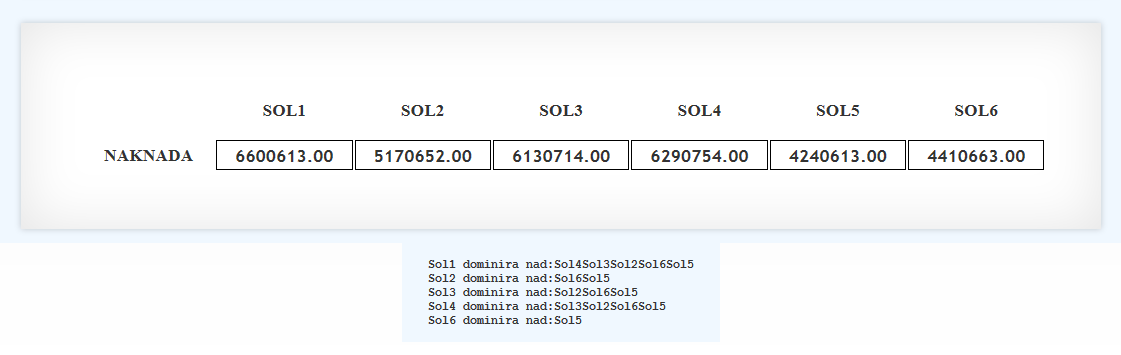
Najbolja je očito solucija 1 sa svojih 6M+. To postaje jasno već nakon 2-3 pokretanja programa.

Isječci ispisa pokazuju višestruke rezultate. Ostavljeno je tako da se vidi generiranje u različitim generacijama te učinci mutacije.

Uglavnom, sve navedeno u I Zaključku po primjeru iz III poglavlja, najjednostavnije je za zaključiti, jednostavno se ne odnosi na ovo stanje. Kao da smo na nekom sasvim drugom zadatku. Po zadanim parametrima jasno je što je najbolji izbor te je vidljiv i brojčani odnos prema ostalim solucijama.

Radi tako jasna „odnosa snaga“ mogu se dodati „granične“ solucije kako bi se i njihov međuodnos učinio jasnijim.

Kada se korak-po-korak prođe navedeni put (Suradnici - Stanje koda - Uprava - Lokacije - Trajanje - Održavanje - Informacije - Remote - Rok - Naknada) dobiju se i vrijednosti ostalih solucija:



Slika 35. ES software – rezultati genetskog algoritma

Tek se sada može uspoređivati solucije. Za neke je svrhe možda interesantna i frekvencija rezultata u nekom intervalu. Na primjer, možda je rekorder SOL1 tek par puta prešao 4M a posljednji SOL5 većinu stanja ima preko 3M. To bi moglo biti značajno kada bi se po tom modelu ponašao neki proces kojemu je vjerojatnost da će pogoditi put od SOL1 vrlo malena, ali bi značajan broj pokušaja mogao biti veći od bolje rangiranih solucija po kriteriju maksimalne vrijednosti.

# EVEN SWAPS algoritmi i oblici

U poglavlju 4. su zamjene riješene uz pomoć GA. Sada će se pokazati algoritam opgodan za dirketno izračunavanje manjih kaskada. Također se može upotrijebiti i za GA varijantu kada se radi o ravnomjernim promjenama.

Oblici (eng. pattern) su samo spomenuti kako bi se moglo razmišljati o zamjenama na jedan drugačiji način. I ta je tema vjerojatno prilično opširna.

## 5.1 Najveća kaskada

Algoritam za pronalazak najveće kaskade:

1. Koeficijenti se sortiraju po veličini te po umnošku sa svojim atributom.
2. Počinje se od najvećeg koeficijenta prema manjem provjeravati postoji li veza među njima. Ako ne postoji onda se zamjeni sa prvim sljedećim pa se ponovi ovaj korak.
3. Postoji li raspored koeficijenata koji zadovoljava korak 2. :
4. Sprema se za ponovnu upotrebu u sljedećim solucijama.
5. Provjerava se utjecaj veličine atributa na ukupni umnožak. Po potrebi se napravi intervencija u raspored koeficijenata.
6. Ne postoji li raspored koji zadovoljava korak 2. , postupno se dodaju najveći među koeficijentima manjim od jedan pa se ponavlja korak 2.

Jednom kada je napravljen drugi korak za sljedeće solucije se upotrebljava samo 3b. Ako taj, potencijalno najbolji, raspored ne odgovara konkretnoj soluciji, onda se prilagodi posebno za nju.

## 5.2 Optimalan raspored

Sortirani koeficijenti u primjeru idu ovako:

20,20,20,10,10,10,10,10,10,10,10,10,10,10,10,10,5,5,5…

Bilo bi daleko lakše kada bi svi bili različiti, no za očekivati u praksi su upravo ovakve situacije.

Dakle, koeficijent 20 se odnosi samo na Trajanje i ne može se „upotrijebiti“ dva puta jer se radi o jednom atributu i tri „izlaza“ s njega. No desetki ima više i treba napraviti takav raspored da ih bude što je moguće više.

Može se krenuti sa Trajanja jednim od tri moguća puta pa pogledati koji će najdalje dogurati. To bi značilo da je potrebno proći sve tri mogućnosti, a možda se uopće ne može složiti raspored polazeći od Trajanja.

Zato je bolje organizirati traženje najdužeg puta za skup atributa koji imaju koeficijent 10 pa pogledati odgovara li nekom od putova koji dolaze sa višeg koeficijenta. Kako se može dogoditi da najduži put nije i najbolji jer možda nema nastavka prema nižim koeficijentima, potrebno je složiti sve moguće putove za dani broj koeficijenata. Posao kao rođen za prolog…

Jedan od putova:

Rok ->Naknada->Remote

Postoji i duži put, Uprava->gornji slijed, ali Uprava nema vezu na koeficijent višeg reda.

Prvih pet atributa, gledajući s kraja, s desna, bi dakle bili:

Trajanje-> Rok ->Naknada->Remote

No Remote nema u listi koeficijenata skroz do koeficijenta 1, dakle preskočili bi 5,4,2,… međutim Remote, iako pod koeficijentom 1, može ići na Održavanje a sa njega je moguće pod koeficijentom 10 ići na Naknadu, zato bi to trebalo malo preurediti:

Trajanje-> Održavanje->Informacije-> Remote->Rok->Naknada

Dakle ovo:

10x1x5x10x20xNaknada+1x5x10x20xRok+5x10x20xRemote+10x20xInformacije+20xOdržavanje+Trajanje

Treba usporediti s ovime:

Trajanje-> Rok->Naknada-> Remote-> Održavanje->Informacije

10x1x10x10x20xInformacije+1x10x10x20xOdržavanje+10x10x20xRemote+10x20xNaknada+20xRok+Trajanje

10000xNaknada+1000xRok+1000xRemote+200xInformacije+20xOdržavanje+Trajanje

20000xInformacije+2000xOdržavanje+2000xRemote+200xNaknada+20xRok+Trajanje

Jedinična vrijednost prvog rasporeda bi bila 12221 a drugog 24221; uzevši da je vrijednost svakog atributa jedan dolazimo do mjerila kvalitete pojedinog rasporeda.

10x1x5x10x20x1+1x5x10x20x1+5x10x20x1+10x20x1+20x1+1

10x1x10x10x20x1+1x10x10x20x1+10x10x20x1+10x20x1+20x1+1

Razlikuju se u 4. iteraciji:

5x10x20x1 =1000

10x10x20x1 =2000

Da su daljnje iteracije jednake:

… NxMx1000 + Mx1000 + 1000

… NxMx2000 + Mx2000 + 2000

… NxMx(1000+1000) + Mx(1000+1000) + (1000+1000)

Razlika bi bila:

1000 + Mx1000 + MxNx1000…

(10-5)x200+ Mx(10-5)x200 + MxNx(10-5)x200…

[ (10-5) + Mx(10-5) + MxNx(10-5)…] x200

Nivelacija na niže u prvoj sljedećoj iteraciji:

k2 – koeficijent kod prekida drugog slijeda

k1 - koeficijent kod prekida prvog slijeda

I - ukupna vrijednost do točke prekida

M,N… - koeficijenti koji se nastavljaju od prekida

… NxMx1000 + Mx1000 + 5x200

… Nx[ M/(10/5)] x2000 + { [ M/(10/5)]x2000 - (10-5)x200} + 10x200

{ [ M/(10/5)] x 2000 } - (10-5)x200

Redukcija novog zadnja razlika radi

Koeficijenta iteracija promjene koeficijenta

Prvi član svodi novi umnožak na gornju vrijednost dijeleći novi koeficijent količnikom prethodna dva. Drugi član oduzima višak vrijednosti iz prethodne iteracije.

{ [ M/(k2/k1)]xk2xI } – (k2-k1)xI

M x k1 x I - (k2-k1)x I

I x (M x k1 +k1 –k2)

Za podatke iz primjera , k1=5 i k2=10, ako je M=1 te iteracije uopće neće biti: vrijednost iteracije se poklapa sa naknadom:

… NxMx1000 + Mx1000 + 1000

… NxMx(5/10)x2000 + 0 + 2000

Razlika prema originalnom slijedu:

[  (k2-k1) +Mx(k2-k1) +MxNx(k2-k1)…] x I

Ukupno reducirani slijed nakon redukcije :

[ (M x k1 +k1 –k2)+ MxNx(k1/k2) + MxNxKx(k1/k2)…] x I

Nivelacija na niže u drugoj iteraciji:

… NxMx1000 + Mx1000 + 5x200

… + { [ N/(10/5)]x2000 - (10-5)x200 - Mx(10/5) } + Mx2000 + 10x200

… + { [ N/(k2/k1)]xk2x200 - (k2-k1)x200 – Mx/(k2/k1) xk2x200 } + Mx2000 + 10x200

+ { Nxk1x200 - (k2-k1)x200 – Mxk1 x200 } + Mx2000 + 10x200

+[  Nxk1- (k2-k1) – Mxk1] x I+ Mx2000 + 10x200

+[ ( N-M)xk1- k2+k1] x I+ Mx2000 + 10x200

Razlikuje se od prethodnog slučaja samo u oduzimanju prvog koeficijenta. Dakle:

+[ (K- N-M)xk1- k2+k1] x I+ NxMx2000 + Mx2000 + 10x200

Ukupno reducirani slijed nakon redukcije :

[ ( (N-M) x k1 +k1 –k2)+ KxNx(k1/k2) + GxNxKx(k1/k2)…] x I

## 5.3. Nivelacija na više:

Hajdemo vidjeti što se dešava kod povećanja :

… NxMx2000 + Mx2000 + 2000

… NxMx1000 + Mx1000 + 1000

… NxMx(1000+1000) + Mx(1000+1000) + (1000+1000)

Razlika je ista:

1000 + Mx1000 + MxNx1000…

(10-5)x200+ Mx(10-5)x200 + MxNx(10-5)x200…

[ (10-5) + Mx(10-5) + MxNx(10-5)…] x200

[  (k2-k1) +Mx(k2-k1) +MxNx(k2-k1)…] x I

Nivelacija na više u prvoj sljedećoj iteraciji:

… NxMx2000 + Mx2000 + 10x200

… Nx[ Mx(10/5)] x1000 + { [ Mx(10/5)]x1000 + (10-5)x200} + 5x200

{ [ Mx(10/5)] x 1000 } + (10-5)x200

Redukcija novog zadnja razlika radi

Koeficijenta iteracija promjene koeficijenta

Prvi član svodi novi umnožak na gornju vrijednost množeći novi koeficijent količnikom prethodna dva. Drugi član dodaje manjak vrijednosti iz prethodne iteracije.

{ [ Mx(k2/k1)]xk1xI } + (k2-k1)xI

M x k2 x I + (k2-k1)x I

I x (M x k2 +k2 –k1)

U ovom slučaju , k1=5 i k2=10, ako je i M=1, redukcijski član sa nadoknadom je 3000:

… NxMx2000 + Mx2000 + 2000

… NxMx(10/5)x1000 + 3000 + 1000

Razlika prema originalnom slijedu:

[  (k2-k1) +Mx(k2-k1) +MxNx(k2-k1)…] x I

Ukupno reducirani slijed nakon redukcije :

[ (M x k1 +k1 –k2)+ MxNx(k2/k1) + MxNxKx(k2/k1)…] x I

## 5.4. Promjena rasporeda nivelacijom

Kada se utvrdi da je potrebna promjena rasporeda, to se može napraviti na sljedeći način:

1. Nivelacija na više/niže ne mjestu prekida (ulaz)
2. Utvrđivanje razlike do mjesta drugog prekida (izlaz)
3. Nivelacija na niže/više ne mjestu prekida (izlaz)
4. Nivelacija je uspješna ako se reducirane vrijednosti poklapaju, odnosno ako je redukcija izlaza >= ulazu

Sada treba utvrditi kada je promjena rasporeda moguća.

## 5.5. Kriteriji za promjenu rasporeda

Idemo pogledati razliku dva slijeda koji predstavljaju dva moguća puta između A i B:

A x kAB + B - (B x kBA + A) = A( kAB- 1) +B(1 - kBA)

1. kAB=1

kBA=1

Ako su koeficijenti jednaki jedan radi se o zbrajanju pa su dva slijeda ista: A+B=B+A

1. kAB=1/2

kBA=2

Koeficijenti različiti od jedan znače i sigurnu razliku u slijedu:

A( kAB- 1) = - B(1 - kBA)

A( kAB- 1) = - B x kBA(kAB-1)

A= - B x kBA

Kada su i koeficijenti i vrijednosti atributa uvijek pozitivni, ovakav slučaj se ne može desiti.

1. Budući da ta razlika, osim kod trivijalnog kXY=1 uvijek postoji, u kojem slučaju je zbroj veći? Recimo da je:

A( kAB- 1) +B(1 - kBA) > 0

A( kAB- 1) + B x kBA(kAB-1)> 0

(kAB-1) [ B x kBA + A] > 0

Dakle, prvi slijed je veći od drugog samo ako je koeficijent prvog slijeda veći od jedan.

[ AB] > [ BA] ako je kAB>1.

Stvari se kompliciraju sa trećim atributom:

( [ A x kAB + B ]kBC +C)- ( [ B x kBA + A]kAC +C) =

A( kAB x kBC- kAC) +B(kBC – kBA x kAC)

Sljedovi su isti kada je gornji zbroj nula, a to se može desiti samo u slučaju da su svi koeficijenti jednaki jedinici: A+B+C = B+A+C

Prvi je veći od drugoga ako:

A x kAB x kBC +B x kBC > A x kAC +B x kBA x kAC / kAB x kBC

A + B x kBA > (A + B x kBA) [ kAC/ (kAB x kBC) ]

1 > kAC/ (kAB x kBC)

kAB x kBC > kAC

[ ABC] > [ BAC] ako je kAB x kBC > kAC

Još je važna i permutacija [ ABC] > [ CAB]

A x kAB x kBC +B x kBC + C > C x kCA x kAB +A x kAB + B

kBC(A x kAB +B + C x kCB) > kAB (C x kCA +A + B x kBA)

kBCxS[C] > kAB xS[A]

Dakle [ ABC] > [ CAB] ako je

kBCx(zvijezda oko C) > kAB x(zvijezda oko A)

Ako sljedove sa tri atributa proširimo na 4 tako da se nastavljaju na istu točku, rezultat je isti kao da tog nastavljanja nije bilo.

( [ A x kAB x kBC + B x kBC +C]kCD + D)-

( [ B x kBA x kAC + A x kAC +C]kCD + D) =

A( kAB x kBC x kCD – kAC x kCD) +B(kBC x kCD – kBA x kAC x kCD)

Prvi je veći od drugoga ako:

A x kAB x kBC x kCD +B x kBC x kCD > A x kAC x kCD +B x kBA x kAC x kCD

A što predstavlja istu jednadžbu iz prethodnog slučaja sa 3 atributa pomnoženu sa kCD. Zato ćemo je i podijeliti sa / kAB x kBC x kCD i dobiti

A + B x kBA > (A + B x kBA) [ kAC/ (kAB x kBC) ]

1 > kAC/ (kAB x kBC)

kAB x kBC > kAC

Dobilo se isto što i u slučaju tri atributa jer se sljedovi ne razlikuju u putu prema četvrtom atributu:

[ ABCD] > [ BACD] ako je kAB x kBC > kAC

Općenito se četiri atributa rješavaju ovako:

[ ABCD] > [ ACDB]:

A x kCD x kAB x kBC + B x kBC x kCD + C x kCD +D

>

A x kCD x kAC x kDB +B + C x kCD x kDB + D x kDB

Izvući ispred zajedničke koeficijente, kCD i kDB:

kCD x (A x kAB x kBC + B x kBC + C +D x kDC)

>

kDB x (A x kCD x kAC +B x kBD + C x kCD + D)

Razlika između C-zvijezde i vrijednosti u zagradi prvog retka:

A x kAB x kBC +B x kBC +C +D x kDC- (A x kAC + B x kBC +C +D x kDC)=

A x (kAB x kBC – kAC)

Dakle:

kCD x (S[C] + A x (kAB x kBC – kAC))

>

kDB x (S[D] + A x (kAC x kCD – kAD))

Razlika prema slučaju sa tri atributa je u članu kojime se odlučuje koji slijed sa tri atributa je veći. Zapravo se nastavlja na prethodni proces odlučivanja. Zvijezde su pogodne jer su zapravo samo zbroj umnožaka atribut x koeficijent i centralnog atributa, a to se lako dobije iz tabele sortiranih koeficijenata.

Svaki se slijed može predstaviti sa trojkama , četvorkama ili bilo kakvim n-torkama:

U x kUA x kAB x kBC x kCD x kDE +A x kAB x kBC x kCD x kDE + B x kBC x kCD x kDE +C x kCD x kDE + D x kDE + E

=C x kCD x kDE + D x kDE + E + kBC x kCD x kDE(U x kUA x kAB +A x kAB + B)

[UABCDE] =[CDE] + kBC x kCD x kDE [UAB]

kCD x kDE se odnose na [CDE] dok je kBC prijelaz između točaka prekida, C i B.

[UABCDE] =[BCDE] + kAB x kBC x kCD x kDE [UA]

[UABCDE] =[DE] + kCD x kDE [BC]+ kAB x kBC x kCD x kDE [UA]

Jednom „sklopljen“ raspored se uvijek može po potrebi prekinuti te rekonstruirati uz nove parametre.

## 5.6. Oblici

Osnovna zamjena:

A x kAB + B

doista ima značajke jednostavne linearne regresije. Koeficijent je i izabran tako željenu promjenu jedne veličine pretvori u drugu. Kada je riječ o jednoj neovisnoj zamjeni regresija nije pogrešna usporedba.

No, čim je više atributa u relaciji više se ne radi o regresiji čak i kada jednadžbe izgledaju vrlo slično regresijskima, na primjer zvijezda:

A x kAD + B x kBD + C x kCD

Ovi koeficijenti nisu rješenja normalnih jednadžbi već pokazuju linearne odnose između varijabli čije ime nose.

Dakle, iako se ne radi o regresiji nije pogrešno, sa stajališta donositelja odluke, razmišljati o zvijezdi kao o matematičkom postupku koji pokazuje kolika je ukupna vrijednost omjera oko centralne varijable zvijezde. Ili kolika je vrijednost svih razmjena koje su vezane za jedan atribut.

O kaskadi možemo razmišljati kao o kontejneru koji akumulira sve što dođe do njega.

U označavanju slijeda zamjena ide se od početnog atributa prema zadnjem: [ABCDE]

Kaskada: C, C[ABC] govori da se u prvom slučaju radi kaskadi svih atributa koja završava na C, a u drugom slučaju da u toj kaskadi učestvuju samo tri atributa.

Zvijezda: S[ABC], S[C\*] ,S[C\*]; prvi slučaj je zvijezda koja završava u C, drugi slučaj je zvijezda svih članova koja završava u C te treći zvijezda od onih članova koji preostanu od drugih oblika te koja završava u C.

Zvijezda se može pretvoriti i kaskadu i obratno, no samo u slučaju kada se radi o 3 atributa:

(A x kACxkBC + B x kBC + C ) x kCB

A x kAC + B + C x kCB

Kako je množenje sa kCB isto što i dijeljenje sa kBC postupak pretvorbe kaskada->zvijezda ili obratno je isti, pitanej je samo sa čime se množi odnosno dijeli.

Neovisne zamjene su one zamjene čije varijable nisu učestvovale u prethodnom ciklusu zamjena.

Na primjer:

A x kAB + B

C x kCD + D

Ako sada idemo mijenjati dalje:

(A x kAB + B )xkBD + C x kCD + D

A x kAB xkBD + BxkBD + C x kCD + D

Rezultat nema strukturu kaskade, dva člana imaju po jedno množenje manje.

Oblici se mogu kombinirati. Mogu biti irelevantni pa će se pomnožiti koeficijentom kompenzacijskog atributa i pribrojiti stanju koje kompenzacijski atribut već ima. Ili mogu biti kompenzacijski pa će se pomnožiti irelevantni oblik koeficijentom atributa koji taj oblik sadržava te će mu se pribrojiti.

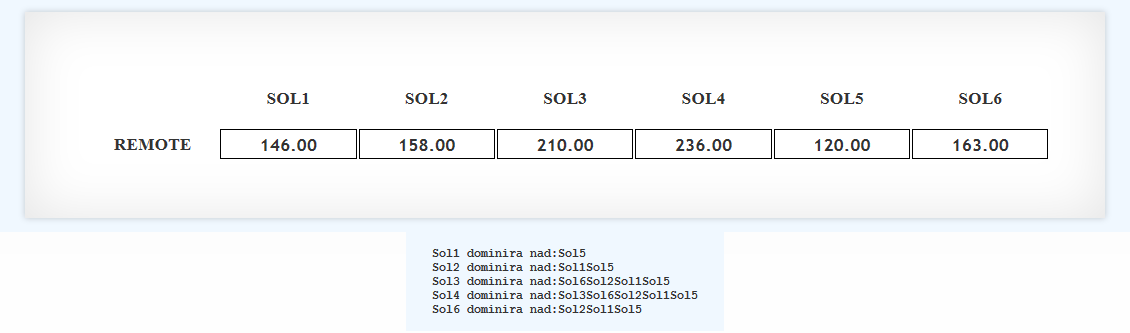
Kod tih kombiniranja treba paziti kakvi su međusobi koeficijenti na sastavu skupina.

To bi otprilike bila umjetnost zamjena: pronaći srodne atribute, grupirati ih po nekom od oblika te onda te oblike povezati. Na taj način zamjene slijede logiku njihove srodnosti onako kako ju je donositelj odluke prepoznao.

## 5.7. III Zaključak po primjeru

Sve što je rečeno o oblicima ima direktnu upotrebu kod planiranja zamjena. Logičke dijelove treba povezati u adekvatne oblike pa onda takve cjeline povezati lijevim ili desnim dodavanjem. Pri tome treba paziti da se koeficijent izmešu pojedinih dijelova izabere obzirom na povezanost cjelina a ne dva predmetna atributa.

Ovako izgleda zvijezda oko Remote:



Slika 36. ES software – oblici na primjeru - zvijezda Remote

Bez velikih množenja opet dolaze do izražaja veličine atributa pa Sol3 i Sol4 opet dominiraju.

Sol1: C/ S[Remote]=45210

1,55

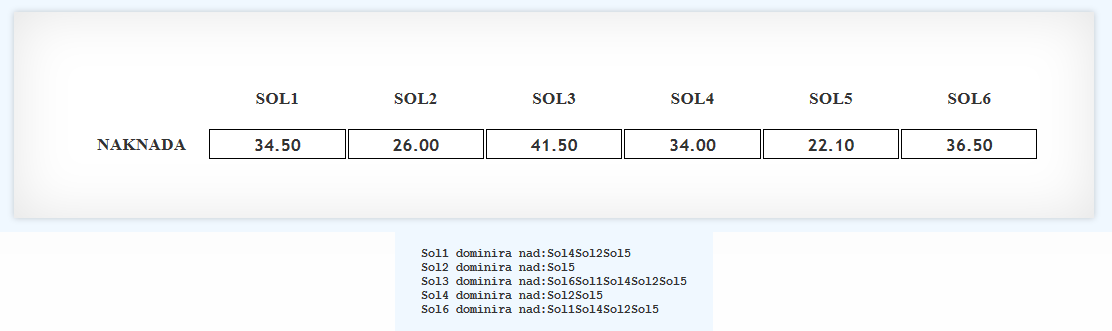
Sol3: C/ S[Remote]=29194

Sol1 C / Sol3 C= 1,077

Sol1: S[Remote] / Sol3: S[Remote]= 0.7

Kaskade Sol1 i Sol3 se razlikuju za cca 8% dok se količnici kaskada i zvijezda oko Remote razlikuju za 55%. Zvijezde se razlikuju za cca 70%.

Broj koji pokazuje koliko zvijezda se nalazi u kaskadi govori o odnosu ukupno akumulirane vrijednosti te vrijednosti koja se odnosi samo na veze jednog atributa sa ostalima. Količnik tih vrijednosti kaže da je Remote u Sol1 55% utjecajniji nego u Sol3. Odnos zvijezda , 70% , također vodi sličnu zaključku.

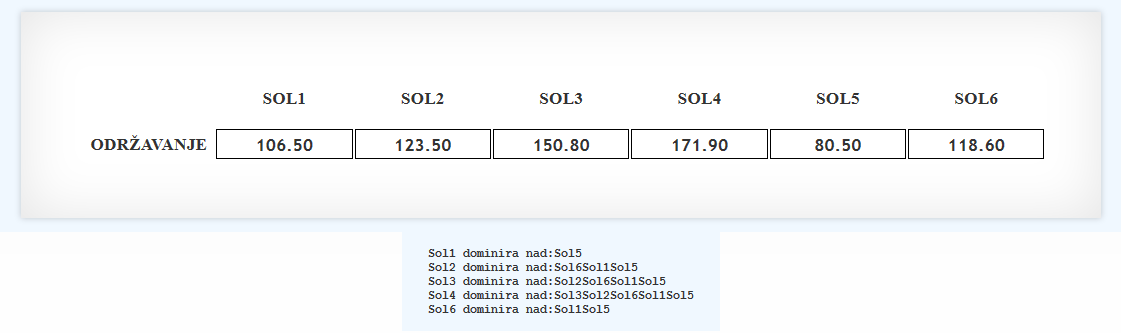


Slika 37. ES software – oblici na primjeru – zvijezda Naknada

Sol1: S[Remote] / S[Naknada] =4,23

Sol3: S[Remote] / S[Naknada] =5,06

Razlika se ne čini značajnom.



Slika 38. ES software – oblici– S[\*]+C[StanjeKoda,Suradnici,Uprava]

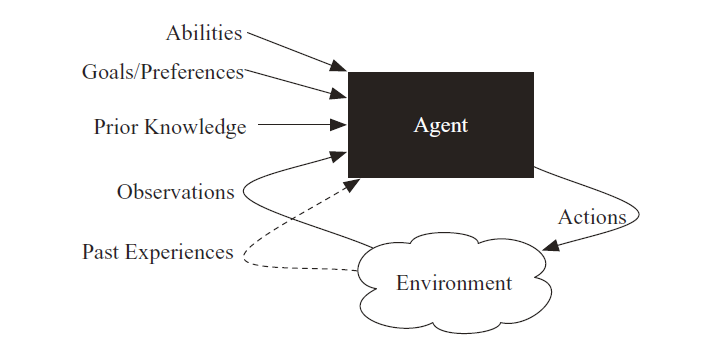
Mala kaskada ne mijenja puno na ukupnom iznosu. Čak su tako pogođeni međusobni odnosi da je zvijezda oko Remote većih iznosa. Prema Naknadi postoje značajnije razlike.

Ako su nekome tri odabrana atributa važniji nego ostatak, ova kombinacija oblika bi mogla biti najbolji izbor. Naravno sa korigiranim koeficijentom između skupina.

# Even Swaps Agenti

## 6.1. Agenti

Prema (Ref.10.):Agent je računalni sustav koji u stanju samostalno djelovati u skladu s ciljevima svojeg korisnika ili vlasnika (pri čemu samostalno određuje načine na koje treba zadovoljiti postavljene ciljeve, umjesto da mu se načini kontinuirano objašnjavaju).



Slika 39. Agent u okružju, Izvor Ref.18

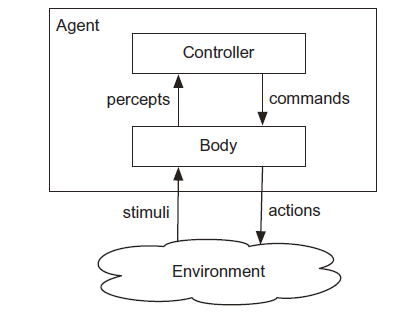
Prema (Ref.11):U bilo kojem trenutku, ono što agent radi ovisi o njegovim:  
• prethodnim znanjima o agentu i okolišu;  
• povijestima interakcije s okolinom koja se sastoji od:

• opažanja o trenutnom okruženju i  
• dosadašnjih iskustava o prethodnim akcijama i promatranjima, ili drugih podataka  
iz kojih se može učiti;

• ciljevima koji se moraju pokušati postići ili preferencija stanja u svijetu i  
• sposobnostima koje su zapravo primitivne radnje koje je u stanju obaviti.

Agenti koji bi koristili even swap mehanizam za primitivnu radnju imaju odgovor na primljene informacije. Njihova je svrha da „daju pravorijek“ na osnovu ulaznih podataka, a kako je kaskada složena na optimalan način, onda je i njen raspored atributa također informacija koja može biti korisna.

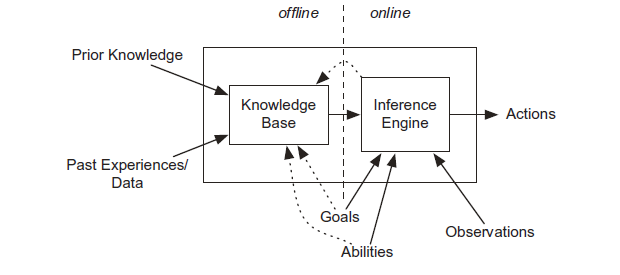
Dva deterministička agenta s istim predznanjem, poviješću , sposobnostima i ciljevima bi trebali raditi istu stvar. Promjena bilo kojeg od njih može dovesti do drugačije radnje.



Slika 40. Agent u okružju, Izvor Ref.19

Agentski sustav se sastoji od agenta i njihovog okruženja. Agent prima podražaje iz okoline, te obavlja radnje u okolišu.

Agent se sastoji od tijela i kontrolne jedinice . Kontrolna jedinica prima percepcije iz tijela i šalje naredbe natrag tijelu.Tijelo sadrži senzore koji pretvaraju podražaja u percepcije te mehanizam koji pretvara naredbe u akcije.



Slika 41. Agent, Izvor Ref.20

Znanje je informacija o domeni koja se koristi za rješavanje problema u toj domeni. Znanje može uključivati opće znanje koje može biti primjenjivo na određene situacije. Dakle, to je općenitiji pojam od uvjerenja o nekom specifičnom stanju . Sustav koji se temelji na znanju je sustav koji koristi znanje o domeni za neku vrstu djelovanje ili za konkretno rješavanje problema.

Filozofi su definirani znanja kao istinito odnosno opravdano vjerovanje. Istraživači umjetne inteligencije imaju tendenciju da izmjenično koriste pojmove znanja i vjerovanja .  
Znanje teži da neku opću informaciju uzme kao istinitu. Vjerovanje je sklonije revidirati neku informaciju na temelju novih informacija. Često uvjerenja dolaze s naznakama u kojoj mjeri im treba vjerovati te modela koji govori o interakciji vjerovanja.

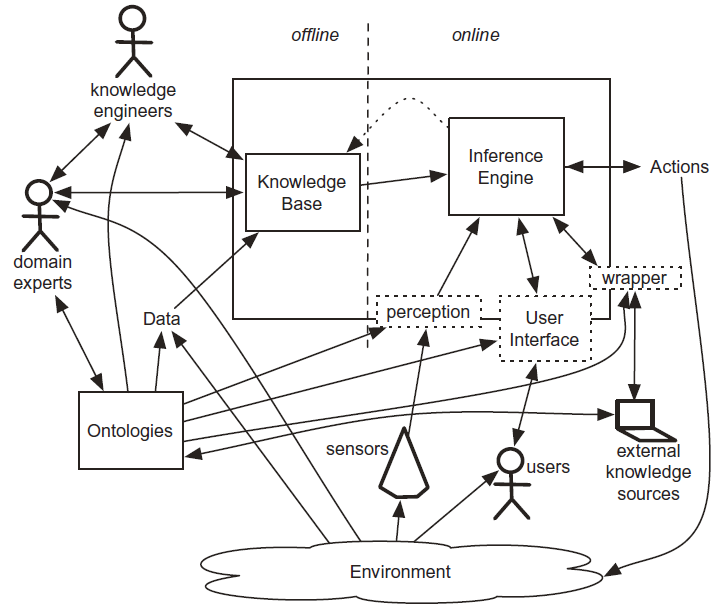
Baza znanja kod even swap agenata je u prvom redu njegova kompenzacijska tabela, a onda i načini zamjena koji bi se trebali primijeniti u pojedinim situacijama. Baza znanja je izgrađena u neaktivnom stanju, a koristi se u aktivnom kako bi učestvovala u kreiranju akcija.

Aktivno stanje se odnosi na vrijeme u kojemu agent djeluje, koristi svoju bazu znanja, promatranja svijeta te ciljeve i sposobnosti da odaberete što učiniti te kako ažurirati svoju bazu znanja. Baza znanja je njegovo dugoročno pamćenje, gdje se čuva znanje koje je potrebno za djelovanje u budućnosti. To znanje dolazi iz prethodnog znanja i kombinira se s onim što se naučilo iz podataka i prošlih iskustava. Stanje uvjerenja je kratkoročna memorija agenta koje održava model u trenutnom okruženju između vremenskih koraka. Ne mora uvijek postojati jasna razlika između općeg i posebnog znanja.

Even swap agent kreiranjem dobije svoju bazu znanja. Ona može vremenom biti mijenjana na nekoliko načina te može biti podijeljena sa drugim agentima. Aktivno stanje znači upotrebu te baze na ulaznim podacima. Sve ostalo se može napraviti u neaktivnom stanju.

Postoji povratna veza između mehanizma zaključivanja i baze znanja, jer se promatranje svijeta te djelovanje u tom svijetu daju podatke iz kojih se može učiti.  
  
Neaktivno stanje, dakle vrijeme prije nego što agent mora djelovati, može se iskoristiti za izgradnju baze znanja koja se koristi u trenutku djelovanja. Uloga neaktivnog stanja je da se pomogne učinkovitosti aktivnog stanja. Baza znanja je izgrađena od predznanja i iz podataka o prošlim iskustvima koja mogu poticati od tog istog agenta ili mu mogu biti dana na korištenje iz drugih izvora. Istraživači tradicionalno drže da slučaj koji uključuje puno podataka i malo prethodnog znanja spada u područje strojnog učenja. Slučaj sa puno predznanja i malo ili nimalo podataka iz kojih bi se učilo razmatra se pod okriljem ekspertnih sustava.

Agentski sustav te okolina koja ga izgrađuje zajedno sa radnom okolinom mogu biti prilično kompleksni. To ovisi od slučaja do slučaja a može se otprilike predstaviti ovom slikom:



Slika 42. Agent u okružju, Izvor Ref.21

Polazište za agente koji bi se bazirali na mehanizmu even swapsa bi trebalo biti prilično jednostavno u njihovoj osnovnoj funkciji; radilo bi se o agentima koji bi imali preciznu i relativno jednostavnu ulogu: na osnovu ulaznih podataka bi trebali dati izlaz, odluku ili sliku stanja svoje okoline. Ako se raspolaže sa dovoljno podataka izlaz bi trebao biti promptan. U suprotnom bi se počelo sa jednim od nekoliko mogućih scenarija (samo)organiziranja u svrhu daljnjeg prikupljanja podataka i njihove obrade što može voditi do situacija koje pokazuje gornja slika a to se ne bi smjelo dešavati na razini jednog agenta iz više razloga. Zato bi trebalo predvidjeti jedan zajednički entitet za sve agente gdje se delegira sve što nije osnovni zadatak even swaps agenta.

## 6.2. Više agentni sustavi

Prema (Ref.10.):Višeagentni sustav je sustav koji se sastoji od niza agenata koji su u međusobnoj interakciji

Prema (Ref.12.): Cilj istraživanja multiagentnih sustava su metode koje nam omogućuju da izgradimo složeni sustav sastavljen od autonomnih agenata koji dok djeluju na razini lokalnog znanja i posjeduju samo ograničene sposobnosti su ipak sposobni za željeno globalno ponašanje.

Želimo znati kako da se opisuje sustav agenata, što on treba učiniti i kako ga podijeliti na ponašanja individualnih agenata. I ono što je najambicioznije je pitanje kako multiagentne sustave osposobiti za reverzni inženjering fenomena kao što su kolonije mrava, gospodarski ili imuni sistem .

Višeagentski sustavi pristupaju problemu pomoću dobro dokazanih alata iz teorije igara, ekonomija ili biologije. To ih nadopunjuje s idejama i algoritmima iz istraživanja umjetne inteligencija kao što su planiranje, metode prosuđivanja, metode pretraživanja i strojno učenje.

Agenti sa mehanizmom even swapsa su u interakciji i sa okolišom te , po potrebi, jedni sa drugima. Osim toga imaju i dodatnu dimenziju, u vezi su sa bazom koja ih kreira i kontrolira. Ta je baza dizajnirana na osnovu ploče (blackboarda) pa će se u daljnjem tekstu tako i referirati.

Posebna kvaliteta je upravo taj reverzni inženjering: kombinacijom even swapsa i prologa u nekim se situacijama može postaviti dinamički model zamjena. Dakle ne samo da agenti mogu dati odluku, što im je osnovna funkcija, već mogu postaviti pravila po kojima donošene. To proizlazi is osobine even swapsa da mu je svako stanje na neki način povezano sa prethodnim, pa ako se utvrdi tijek tih stanja u okolišu može se izvesti pravilo koje ih povezuje.

Ako bi više agenata bilo zaduženo za praćenje različitih stanja u sustavu, onda bi tek interakcijom bilo moguće doći do tih poveznica.

## 6.3. Agenti pri Even swapsu

Kao i uvijek kada se odlučuje o načinu zamjena kod even swapsa, situacija koju modeliramo može diktirati različite oblike i njihove kombinacije. Do sada smo različite situacije predstavljali samo sa zvijezdom ili kaskadom, no tih oblika je puno više (poglavlje u glavi Algoritmi i oblici). Može se definirati agenta koji će različite oblike i njihove kombinacije pratiti neovisno o drugim situacijama (agentima). Radi se o integralnom dijelu programa koji omogućuje definiranje zadataka koji se trebaju izvršiti nad nekim skupom podataka koji se dobiju radom even swapsa.

Na primjer, kod velikog skupa solucija, nakon što ih se sortira upotrebljavajući za tu konkretnu situaciju odgovarajući oblik (pattern) even swapsa moguće je metodama deskriptivne statistike doći do različitih zaključaka.

Moguće je i raditi komparacije između različitih oblika even swapsa.

Sve to je za svaku situaciju drugačije pa bi uz even swaps trebalo napraviti mogućnost definiranja agenata koji se daju „naučiti“ što da prate prilikom izvršavanja even swapsa te što da učine sa tako dobivenim podacima.

Recimo da je interesantno pogledati postotak razlike između maksimalnih kaskada posjednih solucija, mogućnost je koja prva padne na pamet gledajući rezultate even swapa. No, koliki je postotak određene zvijezde u maksimalnoj kaskadi? Prvo treba znati što se hoće postići sa time. Ako je kaskada najveći broj koji se može dobiti iz sustava, a zvijezda broj koji se odnosi na situaciju u kojoj se sustav promatra samo kroz jedan atribut, onda je omjer broj koji pokazuje „snagu“pojedinog atributa u određenoj soluciji.

Da tu ne bi bilo zabune potrebno je prvo utvrditi omjer

Taj broj NIJE jedan što bi se bi se intuitivno možda pomislilo prije nego bi se sjetili da se zvijezda gradi aditivno a kaskada množenjem.

Dakle, tek je omjer gornjeg broja i omjera pojedine zvijezde sa kaskadom relevantan pokazatelj odnosa pojedinog atributa i solucije.

Iz toga se zaključuje da se onda svi ti odnosi po solucijama i atributima mogu staviti u relaciju koja pokazuje neku vrstu veze solucije i atributa.

Koliko je dakle u ukupnom poretku bitan pojedini atribut kod konkretno definiranih solucija? U kojem postotku se to mijenja ako se pojedini atribut promijeni za 1%? Ova ovako postavljena analiza osjetljivosti je posredna veličina koja pokazuje promjenu atributa preko odnosa zvijezda/kaskada. Može se to napraviti i direktno se vežući za zvijezde, kaskade ili bilo koji drugi oblik.

Sve to, kao i prvenstveno činjenica redoslijeda solucija, su polazište za definiranje odgovora agenta pri even swapu, vrlo specifičnog programa kojemu je cilj da prati rad osnovnog procesa i iz njega (i po njemu) izvlači tražene zaključke.

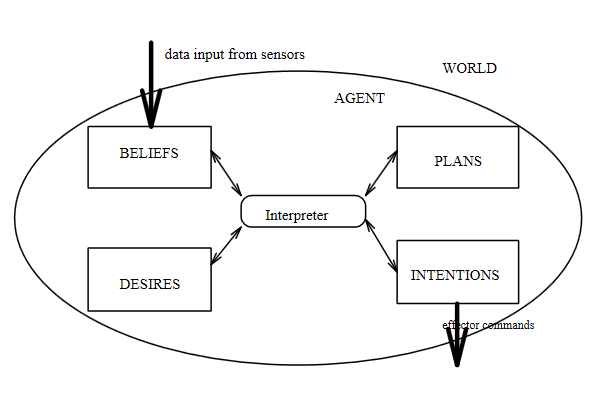
BD agent bi, kako će se vidjeti kasnije, problem udjela zvijezde u kaskadi postavio ovako:

s(X):- X1 is SveZvijezde/Kaskada, X2 is OvaZvijezda/Kaskada, X is X2/X1.

Prolog bi even svapsu bio od velike koristi, no znatno je lakše napraviti neki metajezik posebno u tu svrhu od implementacije prologa.

## 6.4. BDI Agenti

Prema (Ref.13): Možda i najpoznatija agentska arhitektura je Procedural Reasoning System (PRS , Postupovno Rasuđivanje). PRS je primjer od trenutno popularne paradigme beleif-desire-intention (BDI). Kao što slika pokazuje BDI tipično sadrži četiri ključne strukture podataka.



Slika 43. BDI Agent, Izvor Ref.22

Agentova uvjerenja korespondiraju sa informacijama koje agent imao o svijetu koji ga okružuje a koje mogu biti nekompletne ili pogrešne. Uvjerenja mogu biti jednostavna , na primjer varijable u nekom programu. BDI implementacija ih tipično predstavlja simbolično,slično činjenicama u prologu.

Agentove želje intuitivno odgovaraju zadatku na kojemu tog trenutka radi. Za razliku od ljudi kod kojih ne mora biti čvrstog pravila, agentove želje su logički konzistentne.

Agentove intencije reprezentiraju želje koje nastoji izvršiti. U principu agent neće uvijek moći izvršiti sve svoje želje iako su konzistentne. Zato mora odabrati set izvršivih želja kako bi mogao obaviti zadatak. Te odabrane želje su intencije. Agent će nastojati izvršiti intenciju dok god misli da je to moguće a da još uvijek nije izvršeno ili dok se ne uvjeri da je to nemoguće.

Planovi su „recepture“ koje specificiraju akcije koje treba slijediti kako bi se postigao cilj, odnosno izvršila intencija. To proceduralno znanje (know-how) se sastoji od dva dijela: tijelo, ili program, koji definira prirodu akcija te deskriptor koji se odnosi na okolnosti pod kojima plan može biti upotrijebljen (pre-condition) ili koje intencije plan može koristiti (post-condition).

Interpreter ažurira uvjerenja na osnovu opservacija svijeta u kojem se nalazi, generira nove želje na osnovu novih uvjerenja te od njih selektira novi set intencija. Interpreter također selektira akciju koju treba poduzeti na osnovu kurentne intencije i proceduralnog znanja.

Dijagram korak-po-korak even swapsa kroz BDI paradigmu :

* Baza znanja predstavlja uvjerenja. Već je pokazano da njena kompenzacijska tabela može biti predstavljena kao baza prologovih činjenica.
* Solucije su želje. One su mogućnosti od kojih se treba odabrati rezultat. Set solucija je fiksan za jedan ciklus rada agenta i ne izvodi se iz uvjerenja ili vanjskog svijeta, pa je to važan moment koji bitno pojednostavljuje model agenta i njegovu arhitekturu.
* Jasno je da se Izbor solucije, rezultat, odnosi na intencije.
* Promjene su planovi. Mogu se odnositi na bilo koji oblik even swapsa.
* Ciklus kopiranja Solucije u Izbor se dešava po završetku koraka odlučivanja koji je donio neku promjenu. Ako ciklus nije završen promjene se odnose na utvrđivanje dominirane ili praktično dominirane solucije. Ako jeste dovršen i sadrži više od jedne solucije, promjene se utvrđuju numeričkim vrijednostima zadnjeg preostalog atributa. Odabran solucija je ona koja sadrži atribut najveće vrijednosti (ili najmanje, zavisi kako je postavljen zadatak).
* No već kod Agenata pri even swapsu smo vidjeli da to ne mora biti odgovor koji očekujemo od even swapsa. Taj drugi pristup, automatske zamjene, zahtijeva i drugu arhitekturu.

Drugačija arhitektura je potrebna i radi drugačijeg modela rada:

* Automatske promjene trebaju repozitorije „receptura“ za promjene. To je inače uobičajeno u BDI arhitekturi ali su u ovom slučaju ipak nešto drugačije jer se ne odnose na same intencije već na način promjena, dakle obrade ulaznih informacija.
* Osim samih solucija koje su zadane margine buduće odluke, potrebno je pamtiti i brojčane rezultate koji se mogu odnositi na neku situaciju i to pod specifičnom recepturom zamjena.
* Kod slijednog praćenja sustava prethodna stanja determiniraju odluku pa ih je također potrebno pamtiti. Prethodne situacije, dakle kompletan proces donošenja odluke, također mogu biti pamćene u nekoj vrsti kratkoročne memorije. Veći opseg pamćenja se ne bi trebao dešavati na samom agentu jer bi to izašlo iz njegove uske specijalizacije te vodilo kompleksnosti koju bi bilo teže kontrolirati kada je distribuirana po agentima.

A

G

E

N

T

I

AGENT

Izlaz: solucija

Ulaz

Izlaz: raspored

Izlaz: num. vrijednost

P

L

O

Č

A

* Sve što se pamti može biti predmet razmjene sa drugim agentima ili bazom svih agenata. Naime, ukoliko su ulazne informacije nepotpune agent može napraviti tri stvari:
  + Donijeti odluku, ako je moguće, uz zadršku po nepoznatim vrijednostima
  + Iz okoliša nastojati doznati informacije koje mu nedostaju
  + Prijaviti ploči svoje stanje i postati neaktivan

U principu bi se takva analiza trebala odvijati na drugom mjestu kako bi agent zadržao svoju jednostavnost tim više jer bi takav zadatak trebalo obavljati više koordiniranih agenata.

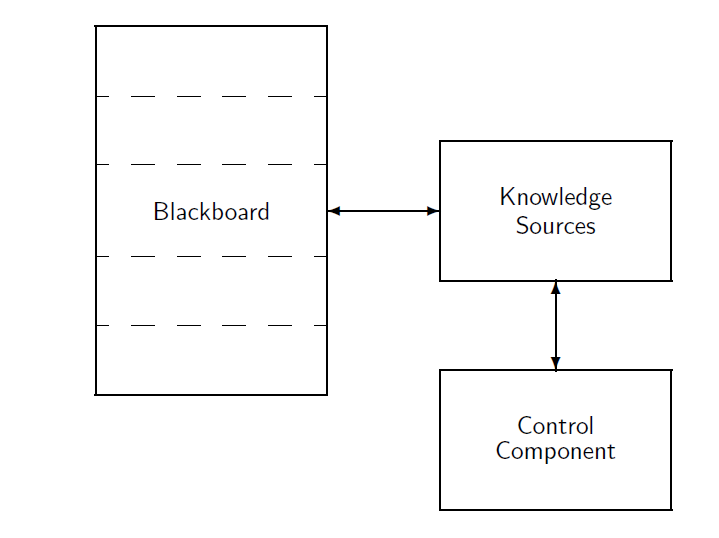
* Nemogućnost agenta da riješi zadani problem prebacuje odluku na zajedničko upravljanje smješteno na ploči. Prema stanju baze znanja i svih agenata ploča može također odustati od problema ili nastaviti proceduru prema svojoj bazi znanja.

## 6.5. Blackboard

Osnovni koncept ploče (blackboard) je da više nezavisnih procesa ažurira ploču kako koji dovrši svoj posao. Klasičan primjer je prepoznavanje govora. Ulazni podatak je zvuk koji će se prepoznati. Audio ulaz se može segmentirati pa više niti može početi tražiti podudaranje isječaka prema pojedinim riječima. Kada pojedina nit pronađe odgovarajuće riječi, ažurirati će ploču s novim stanjem do dosegnute točke. Druga nit može raditi na provjeri gramatike pa sa svoga aspekta opet ažurirati ploču. Desi li se tada da utvrđena riječ ipak nije najbolji izbor, proces se vraća u prethodnu točku i traži se nova riječ koja bolje odgovara utvrđenoj gramatici. Prijedlozi od niti koja je vođena jezičnom bazom znanja postaju polazište za nit koja prepoznaje govor. U načelu je moguće da se tako potvrđena riječ ipak zamjeni onom koja bolje odgovara situaciji…

Prema (Ref.14): Zamislite grupu ljudi, specijalista za svoje područje, koji sjede oko velike ploče. Kada se inicijalni podaci napišu na ploču, svaki od specijalista nastoji upotrijebiti svoje znanje kako bi dao svoj doprinos. Prvi koji dođe na ideju napiše je na ploči omogućujući ostalima da je vide. Proces ide dalje sa svakim od specijalista koji dograđuju postojeće stanje na ploči.

Problem koji se postavlja je povrat u prijašnje stanje nakon uočene pogreške. Specijalisti bi se trebali složiti oko pogrešnog puta, pobrisati ploču te se vratiti na stanje za koje se svi slažu da je zadnje prihvatljivo.



Slika 44. Ploča, Izvor Ref.23

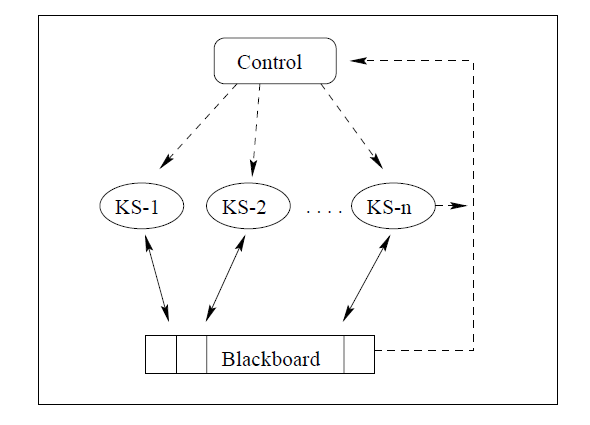
Izvori znanja su neovisni moduli koji sadrže znanje potrebno za rješenje problema. Mogu se u velikoj mjeri razlikovati u reprezentaciji znanja kao i u unutrašnjoj izvedbi. Svaki je neovisan od grupe te ne mora imati nikakva saznanja o ostalima, čak ni postoje li uopće.

Svaki izvor znanja zna točno kada i kako dati svoj doprinos. Za razliku od ekspertnih sistema koju daju pravilo na bazi stimulansa, ovdje se uključuje kompletan modul. Aktivacija pojedinog izvora znanja je njegovo znanje u kombinaciji sa kontekstom budući se svi podjednako bore za resurse, dakle svoju kontribuciju sustavu. Različiti su načini kako se to postiže, od neuronske mreže do neizrazite logike. Izvori znanja su statički repozitoriji dok je njihova aktivacija aktivan proces.

Blackboard je globalna baza podataka koja sadrži sve ulazne podatke, djelomične solucije te ostale podatke od interesa za rješenje problema. Služi i kao komunikacijski medij te privremena pohrana podataka.

Kontrolna komponenta donosi odluke u vrijeme izvođenja programa o samom tijeku rješavanja problema te o optimalnoj upotrebi resursa. Razdvojena je od izvora znanja.

Uobičajeno je mišljenje da je blackboard tek platforma za razvoj na kojoj se testiraju ideje pa se onda iz svega toga razvije jednostavniji sustav. Radi toga se razvija malo takvih sustava pa nema ni komercijalnih programa koji su tome namijenjeni. Penny Nii’s AGE skeletal blackboard framework razvijen na Stanford University jedan je od tri dobavljiva projekta otvorenog koda.

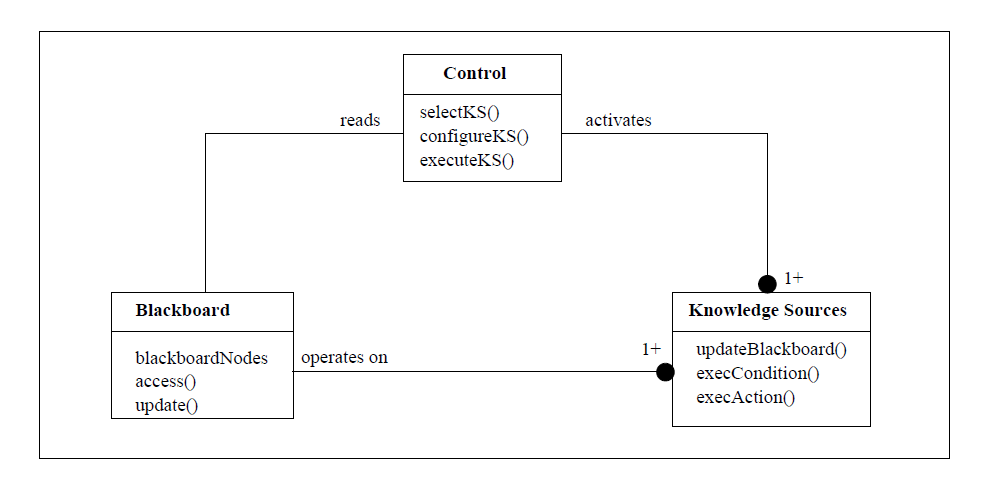


Slika 45. Ploča II, Izvor Ref.24

Prema (Ref.15): Nešto drugačiji pristup pokazuje gornja slika. Ovdje se izvori znanja koriste da riješe manje probleme na koje je podijeljen glavni zadatak. Pišu i čitaju sa ploče dodavajući pomalo relevantne korake koji vode rješenju. Kada je dosegnuta razina značajne promjene generira se događaj. Tu se može nazrijeti nekakva sličnost sa even swapsom.

I ovdje je slična situacija sa angažmanom pojedinog izvora znanja.

Donja slika pokazuje jedan od načina implementacije.



Slika 46. Ploča II, Izvor Ref.25

Prvi korak je definiranje prostora solucija, različitih međukoraka i njihove reprezentacije. Potom se definiraju izvori znanja na koje se odnose pojedini tako postavljeni koraci.

Izvori znanja mogu imati veze sa postojećim modulima. U tom slučaju i ti moduli trebaju slijediti isti princip, biti predstavljeni sustavu na isti način i sa istim sučeljem kao i izvori znanja. To se može postići već i sa jednostavnim umetanjem novog koda (wrapper).

Kontrolna komponenta je uvijek specifična i teško se može naći neki općeniti opis. Ako se radi o formi kompleksnog vremenskog rasporeda onda se često koriste i domenski specifične heuristike koje znaju biti vrlo zahtjevne.

Opisani sustav se koristi za prepoznavanje govora, praćenje i prepoznavanje vozila, identifikaciju strukture molekula proteina, interpretaciju signala sonara, itd…

## 6.6. B(Behavior)D(Decision) Agenti

Even swaps mehanizam bi se mogao iskoristiti za više stvari:

1. Donošenje odluka
2. Usporedba stanja
3. Utvrđivanje poretka
4. Rekonstrukcija nekog stanja u slučaju manjka informacija
5. Izgradnja modela

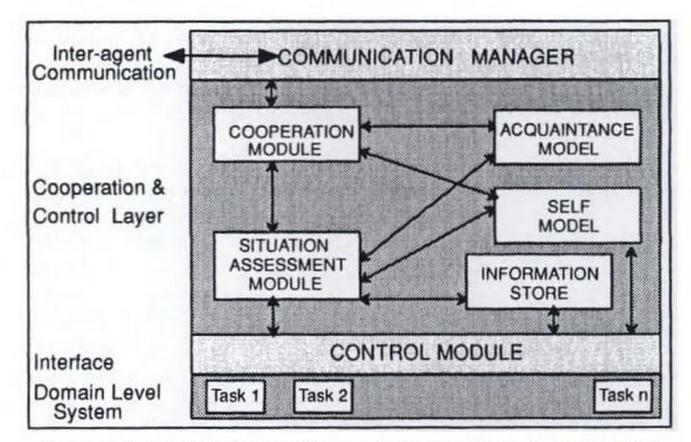
Međutim, sve te mogućnosti pruža isti program, tako da ga je još uvijek moguće zadržati u okvirima jednostavnog i efikasnog.

Kako je even swap prvenstveno namijenjen odlučivanju te svemu što iz toga proizlazi, postavlja se pitanje čemu onda taj bihevioristički dio? Radi toga što se radi o agentu pa za njega vrijedi sve što je rečeno za agente. Da bi imao o čemu odlučivati mora se snalaziti u okolišu. Mora imati znanje koje će ga voditi kroz prikupljanje i distribuciju podataka. Također mora znati što sa samim sobom u odnosu na događaje i njihove posljedice te svoj rad i njegove posljedice. Za ovaj unutrašnji dio BDI već ima veze sa even swapsom pa bi ih trebalo samo prilagoditi situaciji.

Iako , dakle, postoji logička veza prema BDI arhitekturi, koncept even swaps agenta bi trebao biti u dobroj mjeri različit. Uostalom i različite verzije BDI paradigme se bito razlikuju, a u kojoj mjeri to može postati kompleksno pokazuje sljedeći primjer:

# 

Prema (Ref.16): GRATE\* uveliko nalikuje IRMA (Intelligent Resource Bounded Machine Architecture) sa dodatnim komponentama koje se odnose na zajedničko rješavanje problema. Radi se o konceptu grupnih intencija te zajedničke odgovornosti za izvođenje zajedničke aktivnosti.



Slika 47. Ploča II, Izvor Ref.26

Slika također predstavlja BDI arhitekturu što možda nije jasno na prvi pogled.

Kada značaj događaja pokrene novu aktivnost definira se i novi cilj. Zatim se u repozitoriju obrazaca traži adekvatan plan za reakciju na događaj. U prvom redu treba odlučiti može li se cilj zadovoljiti lokalno , dakle na mjestu i od agenta koji ga je preuzeo, ili je potrebna suradnja. Nakon toga se po postojećim intencijama nastoji zaključiti kolika je važnost cilja u odnosu na postojeće aktivnosti.

Interesantna je ovako postavljena arhitektura: događaji, provjera kompatibilnosti i konzistencije se nalaze u Situation assessment modulu. Modul kooperacije nastoji utvrditi potencijalnu vezu sa drugim agentima a Acquaitance modul sadrži posebne mogućnosti (mjesto gdje se sreću znanje i iskustvo) potrebne predmetnoj situaciji.

Najinteresantniji je Self modul: sadrži planove, želje, intencije te zajedničke (po potrebi pridružene) intencije. Kompletan BDI osim baze znanja u jednom modulu. Veza sa uvjerenjima, znanjem, ide preko selekcije događaja ili kompatibilnosti što je mehanizam sličan blackboardu.

Na kraju se zadatak „razbije“ na atomarne dijelove koji se kontrolnim modulom lansiraju i kontroliraju.

Sve je ovo prilično komplicirano pa zato niti razvoj programa nije jednostavan posao.

Prema (Ref.17): AMT (Agent-based Modeling Technique ) predlaže ovakvu vrstu analize:

•Analizirati potrebe sistema

•Konstruirati okvirno BDI agenta

•Identificirati agente i koncepte

•Identificirati relacije između agenata

•Izgraditi scenarije po kojima se ponašaju agenti

•Identificirati ograničenja agenata

Za dizajniranje agenata predlažu sljedeće:

•Napraviti dijagrame relacija agenata

•Napraviti dijagrame interakcija agenata

•Upotrebljavati standardne oblike (patterns)

•Napraviti dijagram komponenti agenata

Osvrt na te prijedloge sa stanovišta jednostavnog (po zadatku a ne nužno i po konstrukciji) agenta od kojega se očekuje, generalno govoreći, reakcija na ulazne podatke:

* U slučaju jednostavnog tipskog agenta analiza sistema ima drugu dimenziju. Usko specijaliziran agent nije prvi izbor za model kompleksnog sustava, eventualno za jedan njegov dio koji odgovara njegovoj specijalizaciji.
* Konstrukcija takvog agenta nije stvar pojedine situacije obzirom na njegovu jednostavnu ulogu. Predviđen je da ga se upotrebljava takvog kakav jeste, no ipak je u jednom dijelu ostavljena mogućnost prilagodbe i dogradnje.
* Koncept agenta je unaprijed poznat tako da ga je moguće uklopiti u različite uloge. Moguće dorade su moguće i dizajnom predviđene, no lokalizirane su samo na zajednički dio te se prvenstveno odnose na različite vrste komunikacije kako unutar populacije agenata potrebne za pojedinu situaciju tako i sa vanjskim svijetom.
* Relacije između agenata su definirane, no agenti ih mogu prilagođavati situaciji.
* Scenariji ponašanja su također unaprijed poznati, „Behavior“ dio koncepta.
* Ograničenja u ponašanju kao i u radnoj funkciji su dio pojedinog programa a implementiraju se stanjem baze znanja.
* Dijagrami relacija i interakcija, u ovako zamišljenoj radnoj okolini, nisu stvar pojedine implementacije. Oni se dešavaju po potrebi , ako su korisniku interesantni, i dio su ugrađene funkcionalnosti agenata.

Takva radna okolina bi se sastojala od dva dijela:

Ploča je modificirani blackboard:

Kontrola agenata

Kontrola veza osim agenata (druge ploče, bufferi…)

PLOČE

Prolog multiheap

datoteke, vanjski izv. mapirani u heap

Repozitorij agenata, solucija, koda, međurezultata

PROLOG MEHANIZAM

* Povrat u prijašnje stanje (backtracking) je dio prolog mehanizma. Putovi i stranputice se rješavaju kao dio prolog programa, dakle automatski.
* Kompenzacijska tabela agenta, njegova baza znanja, je zapravo prologova baza činjenica. Vrlo je jednostavan prijelaz sa jednog oblika na drugi.
* Prolog je i sam po sebi moćan „alat“ za situacije u kojima se traži odluka. Kada rade zajedno, prolog i even swaps mehanizam, sinergija se podrazumijeva. U ovakvoj arhitekturi even swaps agent funkcionira kao predikat sa stanovišta programiranja ploče.
* Umjesto jednog heapa prolog mehanizam je u mogućnosti kreirati ad hoc heap prema potrebama situacije. Umjesto standardne baze blackboarda ovdje se podaci, uvijek kada je to moguće, drže u heapovima. Na taj su način odmah operativni, na raspolaganju prolog mehanizmu.
* Kada bi se držali teoretskih postavki blackboarda, onda bi se sama ploča odnosila na multiheap i repozitorije. Ovdje treba dodati i prolog mehanizam te same ploče koje su organizirane kao tabele. Umjesto globalne memorije i jedne ploče na kojoj se odvija inkrementalno približavanje cilju, ovdje imamo onoliko ploča koliko situacija zahtijeva računajući tu i agregirane ploče koje prate stanje sa više drugih ploča.
* Agent nema čvrstu definiciju već ga ploče kreiraju prema situaciji. Zato se u spremištu nalazi nekoliko verzija koje mogu zatrebati za planiranu situaciju.
* Isto tako ni mehanizam ploča nije čvrsto definiran pa se prilagođava potrebama. Tako da ploča (tabela mapirana u memoriju) može imati svoje agente, a i agent može imati svoju ploču.
* Izvor znanja je ono što se tako definira pri pokretanju programa ili mogu biti dinamički kreirani. Mogu biti i vanjski izvori podataka ( neki web servis, vanjska baza podataka, itd…) .

Izvor znanja uvijek treba biti vezan za predmetnu ploču.

* Agent može biti izvor znanja ako se kreira kao takav.
* Ovako zamišljena ploča nije samo privremeni spremnik međurezultata. Može imati i svog agenta koji je prema svom programu modificira i dijeli neovisno o kontroli.

Osnovni dizajn agenta:

Repozitoriji

Ponašanje

Veza

ES mehanizam, prolog

**BD AGENT**

* ES mehanizam je jedino što mora sadržavati, sve drugo je opcionalno prema situaciji.
* Može sadržavati i prolog mehanizam, na primjer za agente ploča koji onda prologom uređuju tabelu prema logičkim postavkama.
* Repozitoriji sadrže solucije ili numeričke veličine sa kojima se rezultat treba uspoređivati. Također i planove od BDI arhitekture: način na koji se treba mijenjati, oblike ES-a koje treba upotrijebiti u pojedinoj situaciji, itd…
* Veza se odnosi na vezu sa drugim agentima, okolišem ili pločom.
* Ponašanje definira odnos prema okolišu te prema samome sebi, slično Self modulu sa slike (br slike). Organizirano je također na podlozi ploča, na primjer:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A |  |  |  |  |  |
| C |  |  |  |  |  |
| E |  |  |  | R |  |
|  |  | C |  |  |  |
| R |  |  |  |  |  |
|  | F |  |  | Q |  |
| D |  |  |  |  |  |

* Dakle u prvom stupcu fale dva simbola da bi bio popunjen. Kada se oni, tijekom agentova rada dogode, stekli su se uvjeti za promjenu kompenzacijske tabele, vrijednosti atributa, itd… kako promijeniti može biti definirano stanjem ostalih simbola.
* Ploče su tabele mapirane u memoriji pogodne za brzo pretraživanje, ali isto tako i kombiniranje. Mogu se definirati aritmetički ili neki drugi operatori koji rade s pločama tako da se slike stanja ili uvjeta potrebnih za promjene lako mogu provjeriti.
* Na sličan način se agent „ponaša“ i prema okolišu: od koga , kada i kako uzeti podatke, kome dati rezultate… Moguće je zadati ulaz kao izlaz drugog agenta (pipe).
* Agenti su kompleksni onoliko koliko moraju biti. Nema li interakcije sa okolišem (agenti ploče) nema potrebe ni za modul ponašanja. Ako agent samo prati stanje nekog procesa, nema potrebe za repozitorijima, itd…

## 6.7. Kako odlučuje BD agent

Pretpostavimo da Agent ima spremljeno 7 solucija te dopušteni ulaz novih podataka. Iz repozitorija oblika za ovu situaciju je odabrana kaskada pa izlaz može biti:

* Najbolja solucija bez obzira je li to neka od postojećih ili ova upravo zapisana. Nova može biti sačuvana ili odbačena.
* Položaj nove solucije, na primjer 1,3,5,4 - S -2,6,7, dakle bolja je od solucija 1,3,5,4 ali je slabija od 2,6,7.
* Vrijednost kaskade ili nekog oblika nove solucije
* Raspored atributa najveće kaskade.

Ulaz može biti neki broj, tada izlaz može biti:

* Položaj tog broja među vrijednostima kaskada, ili nekih drugih oblika, spremljenih solucija.
* Jednostavno Da/Ne ukoliko je taj broj manji/veći od najmanje/najveće vrijednosti kaskade

Ulaz nadalje može biti raspored atributa u soluciji ali bez svih vrijednosti atributa. Izlaz tada može biti:

* Intervali vrijednosti koje atribut može imati.
* Prijedlog promjene rasporeda prema nekoj od postojećih solucija.

Ulaz može biti i veći broj solucija. Izlaz tada može biti:

* Sortirane solucije prema najvećoj kaskadi ili nekom od oblika.
* Neka statistička obrada nad sortiranim solucijama.
* Raspored atributa najveće kaskade.

Kod agenta koji prati promjene sustava, ulazi su sukcesivne promjene stanja. Agent tada računa, prema zadanim postavkama, vrijednosti atributa ili koeficijenata kojime bi se opisao model kojega prati.

Ako više agenata prati iste promjene, svaki sa svojim postavkama, mogu se dijelovi njihovih baza znanja međusobno mijenjati te na taj način improvizirati genetski algoritam.

Svako stanje agenta se da opisati pločom tako da i ploče mogu biti ulazno-izlazne varijable.

## 6.8. Primjeri

Na nekoliko primjera su pokazani neki od opisanih koncepata.

Primjeri su ilustrativne prirode no do eventualne praktične upotrebe mali je korak.

## 6.8.1 Trgovački putnik

Zadano je 7 gradova. Treba naći raspored posjete svakome od njih tako da se svaki posjeti samo jednom.

Zadatak se može riješiti even swapsom koji ima i koeficijente i atribute postavljene na jedan.

Na raj će se način dobiti kaskada oblika:

…+ 1x1x1x1 + 1x1x1+ 1x1x+1

Crvena jedinica se odnosi na atribut. Rezultat će biti broj atributa no raspored će odgovarati prolazu kroz stablo definirano kompenzacijskom tabelom.

Ako se traži samo raspored može se upotrijebiti i prolog.

Baza činjenica od prologa bi izgledala ovako:

s(G7, G6 ,1):-true.  
s(G6, G1 ,1):-true.  
s(G6, G2 ,1):-true.  
s(G7, G4 ,1):-true.  
s(G5, G3 ,1):-true.  
s(G5, G2 ,1):-true.  
s(G4, G2 ,1):-true.  
s(G3, G1 ,1):-true.  
s(G6, G4 ,1):-true.  
s(G2, G1 ,1):-true.  
s(G6, G5 ,1):-true.  
s(G7, G1 ,1):-true.  
s(G5, G6 ,1):-true.  
s(G3, G6 ,1):-true.  
s(G1, G7 ,1):-true.  
s(G6, G3,1):-true.

Rješenje:

G4-> G2-> G1-> G7-> G6-> G5-> G3

U istom poduzeću rade još dva trgovačka putnika s time što oni stanuju u gradovima G3 i G5.

Postoje li putovi i za njih?

Rješenje: Postoje.

G3-> G6-> G5-> G2-> G1-> G7-> G4

G5-> G6-> G3-> G1-> G7-> G4-> G2

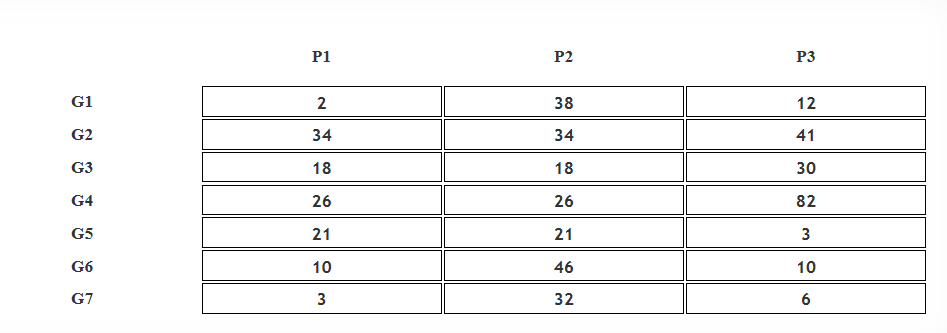
Ta se dvojica mogu sresti u gradu G6. Drugi putovi se ne sijeku.

Hajdemo sada dodati vrijednosti atributima. To bi bio prosječni promet kojega pojedini putnik ostvari po pojedinom gradu. Kaskada će izgledati otprilike ovako:

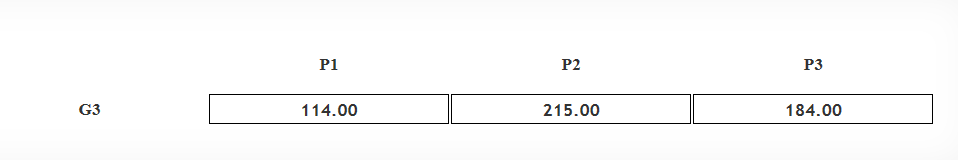
…+ 1x1x1xA + 1x1xB+ 1x1x+C

Rezultat će biti suma atributa neovisno o putu.

Recimo da taj promet po pojedinom gradu izgleda ovako:



Promet na kraju puta:



Prvi putnik je loš prodavač. Idemo vidjeti možda mu logistika bolje ide.

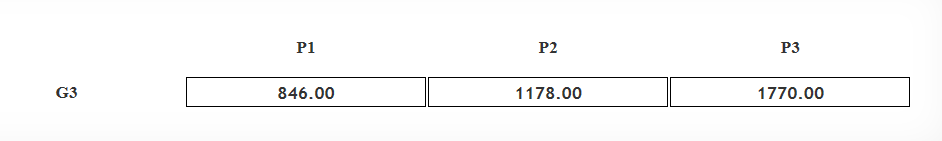
Dodamo li vrijednosti i koeficijentima dobiti ćemo potpunu kaskadu. To možemo tumačiti da pojedini put daje određene prednosti u odnosu na neki drugi put (troškovi puta, dostava, itd…) s time da svaki novi put svojim koeficijentom korigira dotadašnje stanje.

Neka koeficijenti izgledaju ovako:

|  |  |  |
| --- | --- | --- |
| G6 | G1 | 5 |
| G6 | G4 | 4 |
| G2 | G1 | 4 |
| G5 | G3 | 4 |
| G5 | G2 | 3 |
| G7 | G4 | 3 |
| G7 | G1 | 2 |
| G3 | G1 | 2 |
| G6 | G2 | 2 |
| G4 | G2 | 2 |
| G3 | G6 | 1 |
| G7 | G6 | 1 |
| G6 | G7 | 1 |
| G5 | G6 | 1 |
| G6 | G5 | 1 |
| G6 | G3 | 1 |

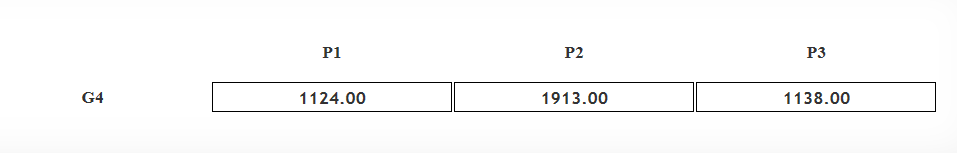
Uz istu tabelu atributa dobiti ćemo ovakvu zaradu:

Put od prvog putnika P1 (G4-> G2-> G1-> G7-> G6-> G5-> G3):



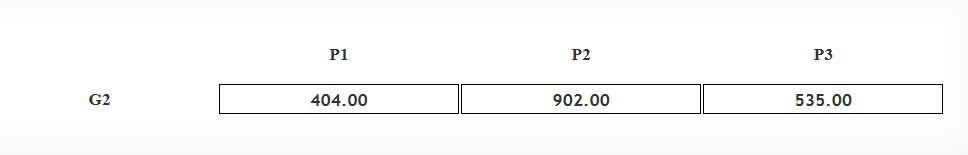
Kada bi istim putom išla sva trojica , u gradu G3 bi se na kraju ispostavilo da je P1 na svome putu najlošiji.

Put od drugog putnika P2 (G3-> G6-> G5-> G2-> G1-> G7-> G4):



I opet je prvi najlošiji a drugi na svojoj ruti dominira.

Put od trećeg putnika P1 (G5-> G6-> G3-> G1-> G7-> G4-> G2):



Ovo je najlošiji put te je prvi putnik također definitivno najlošiji.

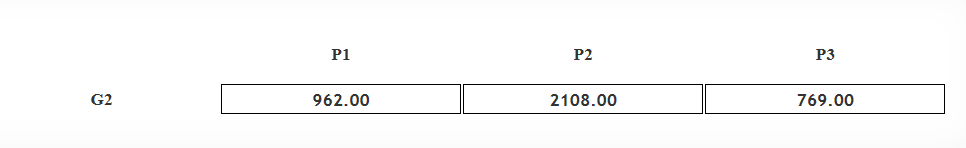
Put je loš jer, na primjer, prijelaz

|  |  |  |
| --- | --- | --- |
| G7 | G1 | 2 |

se ne dešava u tom smjeru već u suprotnom, G1->G7 pa koeficijent nije 2 već ½.

Uprava tog poduzeća bi trebala otpustiti prvog i trećeg putnika a drugog preseliti u grad G6:

G6-> G5-> G3-> G1-> G7-> G4-> G2



Zato što sam napravi prometa bitno više od druge dvojice zajedno.

## 6.8.2 F1- Hamilton i Raikkonen

Problem: kada mijenjati gume? Bolje gume će omogućiti veća ubrzanja i brža kočenja, no vrijeme koje se utroši na zamjenu može biti teško nadoknadivo. Usto bi zamjenu trebalo tempirati u trenutku kada je i konkurencija u istom problemu kalkulirajući sa privremenim gubitkom kako bi se kasnije kapitalizirala prednost. Prerana zamjena će ih opet istrošiti prije kraja utrka, kasnija možda neće ostaviti dovoljno vremena da se tako stečena prednost iskoristi.

Ovdje nije riječ o fizici. Radi se o nizu diskretnih vrijednosti s kojima pratimo stanje. Interesiraju nas njihovi odnosi u nekom malom segmentu puta ili vremena i na osnovu toga stvaramo zaključke. Ti odnosi se mogu fizikalno tumačiti, u nekim situacijama se tako i postavljaju, ali cilj nije rješavanje fizikalnih jednadžbi.

Ubrzanje je prvi pokazatelj kvalitete guma. Uzmemo li jedan zavoj za referentni, ubrzanje nakon svakog kruga možemo predstaviti na ovaj način:

Pribrojak se odnosi na pogrešku ili toleranciju pa ćemo ga radi jednostavnosti zanemariti. Kada bi bio slučajni broj došli bi do teme Markovljevih lanaca.

Prvo što nas interesira su ti koeficijenti koji se dobiju dijeljenjem sadašnje i prethodne vrijednosti ubrzanja. Zatim se usporedi sa situacijom od prošle godine ili sa treninga te se po te dvije skupine podataka izračuna Spearmanov koeficijent korelacije. Po tome vidimo kakva je kompletna situacija u odnosu na referentnu.

Bitna je i usporedba trenutnih stanja pa se prate i prirasti koeficijenata ubrzanja. Razlike dva susjedna koeficijenta možemo iskoristiti za konstrukciju Newtonovog interpolacijskog polinoma, možemo napraviti regresijski model ili na neki drugi način probati pratiti trend promjena. Najbolje bi bilo više stvari pokušati pa kontrolom sa stvarnim vrijednostima vidjeti koji model najmanje griješi.

Zatim je interesantno srednje ubrzanje nakon prolaza promatranog zavoja. Dobiti ćemo ga ovako:

/n = ( a + a + a + …)/n ? =a

A to je n-ti dio kaskade.

Treba ponoviti isto to za ravni dio puta iza zavoja te za usporenje prije sljedećeg zavoja.

Tako smo došli do , i , do tri projekcije trenda te do tri koeficijenta korelacije.

Druga grupa agenata isto to radi sa konkurentom.

Tako dolazimo u situaciju da se na ploči mogu odvijati otprilike ovakva rezoniranja:

uvjet1(XTrend1,\_,\_.\_):- X is Trend1-StvarnoStanje, Y is ABS X, Y < 0.02 , XTrend1 is 1.

uvjet1(\_,XTrend2,\_,\_):- X is KorelacijaA- KonkurencijaKorelacijaA, Y is ABS X, Y< 0.02 , XTrend2 is 1.

…

…

stanje1(X,\_.\_.\_):- X is Vrijeme - KonkurencijaVrijeme.

stanje1(\_,X,\_,\_):- X is X – Konkurencija\_.

…

…

solucije([S1, S2, S3, S4, Sn],Kontrola) :-

S1 = [\_,\_,\_,\_,\_],

S2 = [\_,\_,\_,\_,\_],

S3 = [\_,\_,\_,\_,\_],

S4 = [\_,\_,\_,\_,\_],

Sn = [\_,\_,\_,\_,\_],

S1[XTrend1,\_,\_,\_,\_],uvjet1(X,\_,\_.\_),X> Kontrola, stanje1(XTrend1,\_,\_,\_).

…

Potrebno je doći do situacije za odlučivanje. Naime, vozač ima vrlo malo vremena odlučiti ide li u još jedan krug ili je nastupila optimalna situacija za promjenu guma. Za to mu služe virtualne solucije složene iz prethodnih snimki stanja:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S1 | S2 | S3 | S4 | S5 | S6 | S7 | … | Sn |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| --- |  |  |  |  |  |  |  |  |  |

U trenutku kada je sadašnja solucija bolja od prethodne i zadovoljeni su logički uvjeti, idemo po nove gume.

Ovo je kombinacija realnih pokazatelja (ubrzanja, vrijeme, itd.) te subjektivnih jer osima bolida i njegovih guma u trci učestvuju vozač, staza, vremenski uvjeti, drugi vozači, itd.

## 6.9. IV Zaključak po Primjeru

III Zaključak je pokazao kako već i sama definicija solucija nešto govori. Svaka ima svoj „potpis“ sastavljen od unutrašnjih odnosa.

Sastavljač tih solucija je sastavljanjem već dao neke informacije neovisno o rezultatu izbora. Pitanje je koliko su te informacije konkretno upotrebljive: što bi čovjek mogao napraviti sa odnosom agregiranih veličina subjektivno posloženih u neki , zapravo, slučajni redoslijed jer je on nastao razmatranjem nekog konkretnog posla u nekome trenutku.

Pametni agent bi mogao zaključiti da kaskadno najbolja solucija u odnosu na ostale ima povećan udio Remote zvijezde. Takvih zaključaka bi mogao napraviti desetine te bi dobio neku vrstu slike subjektivnosti koja je takav raspored generirala. Zatim bi u svome svijetu mogao probati naći nešto slično.

Scenarij bi mogao biti otpriike ovakav: poznavajući ključne riječi prema konkretnim podacima od 6 pokušaja može ugrubo filtrirati veliku količinu podataka o ponudi poslova dostupnih na različite načine. Zatim bi takvu selekciju mogao staviti u poznatu situaciju prema stanju kojega ima u memoriji. Negdje će se već naći neka ponuda koja će imati povećan udio Remote zvijezde upotrebljavajući proporcije iz ovih 6 solucija.

# Zaključak

*Reci mi ono što znaš.*

*Zatim mi reci ono što ne znaš.*

*Tek tada mi reci što misliš.*

***Colin Luther Powel****#5*

I

Ovu Powelovu rečenicu bi mogli modelirati ovako:

Odluka= Znanje + Neznanje + Opservacije o obje kategorije (Možda)

Kada bi imali znanje o neznanju bilo bi to također (meta)znanje. Ovako se može desiti da je jedno područje jasno definirano kao nepoznato pa može biti dio određivanja poznatoga.

Nije neobično da se kod odlučivanja koristi i antipod (ELECTRE na primjer) no ovdje bih primijenio jednostavniju metodu: jednostavno bih dopunio sigurno poznati podatak sa onom sigurnošću koja proizlazi iz nedvojbene nesigurnosti

II

Zadavanje atributa i koeficijenata intervalom. To bi bitno podiglo funkcionalnost. Pri tome i rezultat može biti interval s time što se može desiti preklapanje. To je mjesto na kome se može primijeniti neizrazita logika.

III

ES mehanizam može biti upotrijebljen, uz neke promjene, i za Markovljeve lance, zatim modeliranje slijednih situacija (diferencne i funkcionalne jednadžbe), itd…

IV

Sve bi ovo trebalo na isti način ponoviti i sa AHP-om. Dobili bi sinergiju dvije metode koje bi se lijepo dopunjavale.

# Prilozi

## Sortirani koeficijenti

Trajanje Rok 20.000000   
Trajanje Remote 20.000000   
Trajanje Održavanje 20.000000   
Stanje koda Remote 10.000000   
Uprava Rok 10.000000   
Uprava Održavanje 10.000000   
Suradnici Remote 10.000000   
Suradnici Održavanje 10.000000   
Suradnici Rok 10.000000   
Stanje koda Rok 10.000000   
Uprava Remote 10.000000   
Rok Naknada 10.000000   
Naknada Remote 10.000000   
Stanje koda Trajanje 10.000000   
Održavanje Naknada 10.000000   
Održavanje Informacije 10.000000   
Stanje koda Uprava 10.000000   
Informacije Remote 5.000000   
Trajanje Informacije 5.000000   
Rok Informacije 5.000000   
Uprava Informacije 4.000000   
Lokacije Trajanje 4.000000   
Suradnici Informacije 4.000000   
Lokacije Rok 4.000000   
Lokacije Remote 4.000000   
Lokacije Održavanje 4.000000   
Trajanje Suradnici 2.000000   
Naknada Lokacije 2.000000   
Stanje koda Lokacije 2.000000   
Stanje koda Informacije 2.000000   
Naknada Informacije 2.000000   
Suradnici Lokacije 2.000000   
Uprava Lokacije 2.000000   
Trajanje Uprava 2.000000   
Trajanje Naknada 2.000000   
Lokacije Informacije 1.250000   
Rok Remote 1.000000   
Remote Rok 1.000000   
Naknada Stanje koda 1.000000   
Remote Održavanje 1.000000   
Naknada Uprava 1.000000   
Naknada Suradnici 1.000000   
Stanje koda Suradnici 1.000000   
Stanje koda Naknada 1.000000   
Uprava Suradnici 1.000000   
Suradnici Naknada 1.000000   
Suradnici Stanje koda 1.000000   
Uprava Naknada 1.000000   
Stanje koda Održavanje 1.000000   
Održavanje Remote 1.000000   
Suradnici Uprava 1.000000   
Održavanje Rok 1.000000   
Održavanje Stanje koda 1.000000   
Rok Održavanje 1.000000

## Sortirane vrijednosti solucija

Sol1 Suradnici 5.000000   
Sol1 Naknada 3.000000   
Sol1 Stanje koda 3.000000   
Sol1 Informacije 2.000000   
Sol1 Uprava 2.000000   
Sol1 Lokacije 1.000000   
Sol1 Rok 1.000000   
Sol1 Održavanje 1.000000   
Sol1 Remote 0.000000   
Sol1 Trajanje 0.000000   
Sol2 Stanje koda 4.000000   
Sol2 Informacije 3.000000   
Sol2 Lokacije 3.000000   
Sol2 Uprava 3.000000   
Sol2 Naknada 2.000000   
Sol2 Suradnici 2.000000   
Sol2 Trajanje 1.000000   
Sol2 Održavanje 1.000000   
Sol2 Remote 0.000000   
Sol2 Rok 0.000000   
Sol3 Lokacije 7.000000   
Sol3 Informacije 4.000000   
Sol3 Naknada 4.000000   
Sol3 Stanje koda 4.000000   
Sol3 Uprava 3.000000   
Sol3 Suradnici 3.000000   
Sol3 Održavanje 1.000000   
Sol3 Trajanje 1.000000   
Sol3 Rok 1.000000   
Sol3 Remote 0.000000   
Sol4 Uprava 6.000000   
Sol4 Informacije 5.000000   
Sol4 Lokacije 5.000000   
Sol4 Naknada 4.000000   
Sol4 Stanje koda 4.000000   
Sol4 Suradnici 3.000000   
Sol4 Održavanje 1.000000   
Sol4 Trajanje 1.000000   
Sol4 Remote 0.000000   
Sol4 Rok 0.000000   
Sol5 Suradnici 3.000000   
Sol5 Naknada 3.000000   
Sol5 Stanje koda 2.000000   
Sol5 Informacije 2.000000   
Sol5 Lokacije 2.000000   
Sol5 Uprava 2.000000   
Sol5 Održavanje 1.000000   
Sol5 Remote 1.000000   
Sol5 Trajanje 0.000000   
Sol5 Rok 0.000000   
Sol6 Lokacije 4.000000   
Sol6 Naknada 3.000000   
Sol6 Stanje koda 3.000000   
Sol6 Informacije 3.000000   
Sol6 Uprava 3.000000   
Sol6 Suradnici 2.000000   
Sol6 Održavanje 1.000000   
Sol6 Trajanje 1.000000   
Sol6 Rok 1.000000   
Sol6 Remote 0.000000

## Sol1

'Suradnici' => '5'  
'Naknada' => '3'  
'Stanje koda' => '3'  
'Informacije' => '2'  
'Uprava' => '2'  
'Lokacije' => '1'  
'Rok' => '1'  
'Održavanje' => '1'  
'Remote' => '0'  
'Trajanje' => '0'

## Last, middle, first

**Last:**

0 => 'Održavanje'  
1 => 'Trajanje'

**All – Last:**

0 => 'Suradnici'  
1 => 'Naknada'  
2 => 'Stanje koda'  
3 => 'Informacije'  
4 => 'Uprava'  
5 => 'Lokacije'  
6 => 'Rok'  
7 => 'Remote'

**First:**

0 => 'Suradnici'  
1 => 'Stanje koda'

**All-Last-First ->Middle:**

0 => 'Naknada'  
1 => 'Informacije'  
2 => 'Uprava'  
3 => 'Lokacije'  
4 => 'Rok'  
5 => 'Remote'

Fitted middle:

0 => 'Uprava'  
1 => 'Informacije'  
2 => 'Remote'  
3 => 'Rok'  
4 => 'Naknada'  
5 => 'Lokacije'

## Prolog kod

notm*(X,T):-* member*(X,T),!,fail.*

notm*(X,T):-* true*.*

use(member).

t10(Start, X):- t0(Start, X); t1(Start, X).

path(Node, Node, \_, [Node]) *:-* true*.*

path(Start, Finish, Visited, [Start | Path]) :-

t10(Start, X),

notm(X, Visited),

path(X, Finish, [X | Visited], Path).

path(trajanje, uprava, [trajanje], Path).

path(Node, Node, \_, [Node],\_,\_) *:-* true*.*

path(Start, Finish, Visited, [Start | Path],K,S) :-

t10(Start, X,Zz), Tm is K, K is Tm\* Zz,

notm(X, Visited),

path(X, Finish, [X | Visited], Path).

## Prolog listing baze

t0(trajanje, rok):-true.

t0(trajanje, remote ):-true.

t0(trajanje ,odrzavanje ):-true.

t0(stanje\_koda, remote ):-true.

t0(uprava ,rok ):-true.

t0(uprava ,odrzavanje ):-true.

t0(suradnici, remote ):-true.

t0(suradnici, odrzavanje ):-true.

t0(suradnici ,rok ):-true.

t0(stanje\_koda ,rok ):-true.

t0(uprava ,remote ):-true.

t0(rok ,naknada ):-true.

t0(naknada, remote ):-true.

t0(stanje\_koda, trajanje ):-true.

t0(odrzavanje ,naknada ):-true.

t0(odrzavanje ,informacije ):-true.

t0(stanje\_koda ,uprava ):-true.

t0(informacije ,remote ):-true.

t0(trajanje ,informacije ):-true.

t0(rok ,informacije ):-true.

t0(uprava, informacije ):-true.

t0(lokacije, trajanje ):-true.

t0(suradnici, informacije ):-true.

t0(lokacije, rok ):-true.

t0(lokacije ,remote ):-true.

t0(lokacije ,odrzavanje ):-true.

t0(trajanje ,suradnici ):-true.

t0(naknada ,lokacije ):-true.

t0(stanje\_koda, lokacije ):-true.

t0(stanje\_koda ,informacije ):-true.

t0(naknada ,informacije ):-true.

t0(suradnici ,lokacije ):-true.

t0(uprava ,lokacije ):-true.

t0(trajanje, uprava ):-true.

t0(trajanje ,naknada ):-true.

t0(lokacije ,informacije ):-true.

t1(rok ,remote ):-true.

t1(remote, rok ):-true.

t1(naknada, stanje\_koda ):-true.

t1(remote ,odrzavanje ):-true.

t1(naknada ,uprava ):-true.

t1(naknada ,suradnici ):-true.

t1(stanje\_koda, suradnici ):-true.

t1(stanje\_koda, naknada ):-true.

t1(uprava ,suradnici ):-true.

t1(suradnici, naknada ):-true.

t1(suradnici, stanje\_koda ):-true.

t1(uprava, naknada ):-true.

t1(stanje\_koda, odrzavanje ):-true.

t1(odrzavanje ,remote ):-true.

t1(suradnici ,uprava ):-true.

t1(odrzavanje, rok ):-true.

t1(odrzavanje ,stanje\_koda ):-true.

t1(rok, odrzavanje ):-true.

## Prolog listing program

*Path = [trajanje,suradnici,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,remote,odrzavanje,stanje\_koda,lokacije,rok,naknada,uprava]*

*Path = [trajanje,suradnici,remote,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,informacije,remote,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,odrzavanje,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,remote,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,odrzavanje,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,stanje\_koda,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,stanje\_koda,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,odrzavanje,stanje\_koda,lokacije,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,stanje\_koda,lokacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,stanje\_koda,lokacije,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,stanje\_koda,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,rok,naknada,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,naknada,lokacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,naknada,lokacije,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,naknada,lokacije,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,naknada,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,naknada,uprava]*

*Path = [trajanje,suradnici,rok,informacije,remote,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,informacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,rok,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,informacije,remote,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,rok,remote,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,rok,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,remote,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,rok,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,rok,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,informacije,remote,rok,naknada,lokacije,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,informacije,remote,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,informacije,remote,rok,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,informacije,remote,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,informacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,informacije,remote,rok,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,informacije,remote,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,informacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,informacije,remote,odrzavanje,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,informacije,remote,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,informacije,remote,odrzavanje,stanje\_koda,rok,naknada,uprava]*

*Path = [trajanje,suradnici,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,informacije,remote,odrzavanje,stanje\_koda,lokacije,rok,naknada,uprava]*

*Path = [trajanje,suradnici,informacije,remote,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,naknada,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,naknada,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,informacije,remote,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,informacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,informacije,remote,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,remote,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,remote,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,rok,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,rok,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,rok,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,odrzavanje,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,odrzavanje,stanje\_koda,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,remote,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,informacije,remote,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,remote,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,stanje\_koda,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,stanje\_koda,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,stanje\_koda,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,rok,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,rok,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,odrzavanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,odrzavanje,rok,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,odrzavanje,stanje\_koda,rok,naknada,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,lokacije,informacije,remote,odrzavanje,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,naknada,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,lokacije,rok,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,lokacije,rok,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,lokacije,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,lokacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,lokacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,lokacije,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,lokacije,informacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,lokacije,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,informacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,remote,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,remote,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,rok,informacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,rok,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,rok,informacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,rok,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,remote,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,remote,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,odrzavanje,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,odrzavanje,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,informacije,remote,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,informacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,lokacije,informacije,remote,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,informacije,remote,rok,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,informacije,remote,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,informacije,remote,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,odrzavanje,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,odrzavanje,informacije,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,odrzavanje,remote,rok,naknada,uprava]*

*Path = [trajanje,suradnici,stanje\_koda,odrzavanje,rok,naknada,uprava]*

*Path = [trajanje,suradnici,uprava]*

*Path = [trajanje,uprava]*

*Path = [trajanje,naknada,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,remote,rok,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,remote,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,lokacije,rok,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,lokacije,rok,informacije,remote,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,lokacije,rok,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,lokacije,rok,remote,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,lokacije,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,lokacije,rok,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,lokacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,lokacije,remote,rok,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,lokacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,lokacije,remote,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,lokacije,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,lokacije,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,lokacije,informacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,lokacije,informacije,remote,rok,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,lokacije,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,lokacije,informacije,remote,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,informacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,informacije,remote,rok,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,informacije,remote,odrzavanje,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,stanje\_koda,uprava]*

*Path = [trajanje,naknada,stanje\_koda,suradnici,uprava]*

*Path = [trajanje,naknada,uprava]*

*Path = [trajanje,naknada,suradnici,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,rok,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,rok,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,informacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,lokacije,rok,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,lokacije,rok,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,lokacije,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,lokacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,lokacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,lokacije,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,lokacije,informacije,remote,rok,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,lokacije,informacije,remote,odrzavanje,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,stanje\_koda,uprava]*

*Path = [trajanje,naknada,suradnici,uprava]*

*[389 solutions; 291261 inferences; 1.862 cpu; 156.4 klips; heap 565 bytes]*

# 9.Napomene

1. Akronim ELimination Et Choix Traduisant la REalité (ELimination and Choice Expressing REality). Metoda višekriterijskog odlučivanja prvi puta predložena od Bernarda Roya. <https://en.wikipedia.org/wiki/ELECTRE>
2. Akronim **Analytic Hierarchy Process** je također metoda višekriterijskog odlučivanja razvijena od [Thomas L. Saaty](https://en.wikipedia.org/wiki/Thomas_L._Saaty)a 1970. – ih godina. <https://en.wikipedia.org/wiki/Analytic_hierarchy_process>
3. Benjamin Franklin je jedan od osnivača Sjedinjenih Država. Bio je političar, znanstvenik, državnik I diplomat.

<https://en.wikipedia.org/wiki/Benjamin_Franklin>

1. „Quid pro quo“, nešto za nešto, ti meni – ja tebi, odnosi se na situaciju razmjene roba ili usluga u kojoj je transakcija međusobno uvjetovana. U pravnom smislu se koristi kao izraz za konsensualnu svjesnu razmjenu.
2. **Colin L. Powel**, 65. Državni tajnik USA, „Intelligence Reform“, govor 13.09.2004. na Senate Governmental Affairs Committee, <http://www.fas.org/irp/congress/2004_hr/091304powell.html>

**„***Sada, kada mi kažete nešto što će se vrlo vjerojatno dogoditi, tada ja, kao tvorac (službenog) stava, trebam napraviti prosudbu treba li djelovati po tome pitanju ili ne, i neću vas držati odgovornim jer je to prosudba za koju su odgovorni oni koji je stvaraju, a ne obavještajni stručnjaci. Mislim da je ovo pravilo bilo vrlo korisno za mene tijekom proteklih godina: omogućilo je mojim obavještajcima da mi daju činjenice, ali također i da procjene vrlo vjerojatnu pojavu znajući da ja nosim odgovornost za donošenje odluka i kada su na bazi srednjeg ranga informacija o tome što je vrlo vjerojatno da će se dogoditi***“.**

# 10.Reference

1. <https://www.ideapod.com/idea/Ben-Franklins-Moral-Algebra-Electing-between-the-Pro-and-Con/539923ae77656211e4a40300>

2. Even swaps, Teorija odlučivanja, skripta FOI

3. Harvard Business Review, Even Swaps: A Rational Method for Making Trade-off, <https://hbr.org/1998/03/even-swaps-a-rational-method-for-making-trade-offs>

4. Smart-Swaps– A decision support system for multicriteria decision analysis with the even swaps method, Jyri Mustajoki and Raimo P. Hämäläinen ,Helsinki University of Technology, str. 10,[**http://sal.aalto.fi/publications/pdf-files/mmus06b.pdf**](http://sal.aalto.fi/publications/pdf-files/mmus06b.pdf)

5. ODABIR FINALNOG PROIZVODA U PROIZVODNJI ŠLJIVE PRIMJENOM EVEN SWAPS METODE, Josip-Juraj Tominac, SVEUČILIŠTE U ZAGREBU**,** Agronomski fakultet, Studij Agrarna ekonomika

6. An Introduction to Difference Equations, Saber Elaydi, Department of Mathematics Trinity University San Antonio, Texas 78212, str. 3.

7. Prolog, <http://www.enciklopedija.hr/natuknica.aspx?id=50623>

8. Genetski algoritam, Marin Golub, FER, Unska 3, 10000 Zagreb,

<http://www.zemris.fer.hr/~golub/ga/ga.html>

9. Genetički algoritmi, Wikipedia, <https://hr.wikipedia.org/wiki/Geneti%C4%8Dki_algoritmi>

10.Višeagentni sustavi, Markus Schatten, FOI

11. Artificial Intelligence Foundations of Computational Agents, David L. Poole *University of British Columbia*, Alan K. Mackworth *University of British Columbia*, Cambridge University Press

12. Fundamentals of Multiagent Systems, Jose M. Vidal

13. Agent-Based Computing , M.Wooldridge

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.850&rep=rep1&type=pdf>

14. **Blackboard Systems,** Daniel D. Corkill, Blackboard Technology Group, Inc.

15. Thomson-CSF Corporate Research Laboratory, Philippe Lalanda

16. Foundations of Distributed Artificial Intelligence , G. M. P. O'Hare,Nick Jennings,Strana 180, <https://books.google.hr/books?hl=hr&lr=&id=QqJV_hZgCiYC&oi=fnd&pg=PA169&dq=BDI+agent++ordered+triples+(goal,+belief,+reason)&ots=sbASmnucyt&sig=U2a8AWOLIHzw1eHALZZjJIaoYeo&redir_esc=y#v=onepage&q&f=false>

17. Journal of object technology, članak A BDI Agent-Based Software Process

<http://www.jot.fm/issues/issue_2005_11/article3.pdf?q=bdi>

18. Ref. 11. Str. 11.

19. Ref. 11. Str. 45.

20. Ref. 11. Str. 60.

21. Ref. 11. Str. 62.

22. Ref. 13. Str. 7.

23. Ref. 14. Str. 5.

24. Ref. 15. Str. 2.

25. Ref. 15. Str. 4.

26. Ref. 16.