UDC: 62.004.738.5

STRUČNI RAD

PLANIRANJE RESURSA ODRŽAVANJA KORIŠĆENJEM PROGRAMIRANJA SA OGRANIČENJIMA

MAINTENANCE RESOURCE PLANNING USING CONSTRAINT PROGRAMMING

Mr Miodrag Strak, dipl.inž.,

Fakultet informacionih tehnologija, Beograd

REZIME

Donošenje odluka vezanih za organizaciju održavanja, često se odvija u uslovima ograničenog broja vrsta i ograničene količine različitih resursa. Ograničena raspoloživost, takođe, stvara probleme u planiranju aktivnosti, što često dovodi do nepovoljnog raspoređivanja resursa u vremenu. U ovom radu prikazane su mogućnosti primene tehnika zasnovanih na ograničenjima. U toku razvoja ove tehnologije puno se radilo na razvoju i unapređenju sistema zaključivanja. Između ostalog, rezultat tih napora koji se pokazao kao jedan od uspešnijih, jeste uvođenje globalnih ograničenja. Od posebnog značaja je mogućnost implementacije globalnih ograničenja u savremenim objektno-orijentisanim programskim jezicima. U ovom radu razmatra se praktična primena globalnih ograničenja u Java programskom okruženju, za potrebe planiranja resursa održavanja jednog proizvodnog sistema.

Ključne reči: *programiranje sa ograničenjima, alokacija resursa, održavanje*

ABSTRACT

Decision making related with the maintenance planning is often performed in conditions of limited number of types and limited quantity of different resources. Limited availability, also, creates problems in activity planning which often leads to unfavorable resource allocation in time. In this work, application possibilities of constraint programming are shown. During the development of this technology, a great effort was put into the development and enhancing of the reasoning system. The result of this effort, which turned out to be one of the most successful, was introduction of the global constraints. The possibility of implementation of the global constraints in object-oriented programming languages is especially significant. This work deals with practical applications of global constraints in Java programming environment for the needs of resource planning for maintenance of a production system.

Key words: *constraint programming, resource allocation, maintenance*

1. UVOD

Dobro organizovano održavanje u proizvodnoj organizaciji ima značajne ekonomske efekte. Od pravilnog planiranja organizacije, ostvarenja i izvršenja održavanja, zavise i ekonomski rezultati preduzeća, zbog smanjenja investicija za nabavku nove opreme, uz garantovanje dužeg i racionalnijeg iskorišćenja postojeće opreme [1].

U eksploataciji industrijskih postrojenja u našoj zemlji, česta je pojava da, i pored dobro zamišljenih projekata i uspešne realizacije, dolazi do zastoja, pre svega zbog lošeg planiranja održavanja.

Održavanje se, najčešće, sastoji od niza međusobno zavisnih aktivnosti. Osim toga ovu vrstu održavanja prati i veliki broj ograničenja, ali i zahtevi za što efikasnije korišćenje potrebnih resursa. To od onoga koji organizuje proces održavanja zahteva poznavanje niza oblasti: opreme, procesa proizvodnje, prilika u pogonu i sl.

Opšti problem organizacije podrazumeva utvrđivanje dopustivog redosleda aktivnosti, po kome predviđeni poslovi zauzimaju raspoložive resurse. Potrebno je izvršiti izbor između alternativnih planova i resurse dodeliti aktivnostima, vodeći računa o zavisnostima između aktivnosti i ograničenjima koja se tiču ukupnog kapaciteta resursa. Slobodni resursi se mogu dodeliti aktivnostima čije je vreme trajanja unapred definisano, ali tako da nijedan resurs ne bude angažovan za dve aktivnosti istovremeno [2].

Za nesmetano odvijanje procesa održavanja angažuju se svi raspoloživi resursi. To mora biti praćeno dobrom organizacijom rada, naročito kada su isti resursi potrebni i za druge aktivnosti, npr. za potrebe redovne proizvodnje. Zbog, neretko, teških uslova proizvodnje, resursi predviđeni i korišćeni u jednom ciklusu, nedostupni su za korišćenje u sledećem ciklusu procesa redovnog održavanja.

U ovom radu razmatra se mogućnost korišćenja metoda programiranja sa ograničenjima za rešavanje ovog problema. Posebna pažnja posvećena je primeni metoda programiranja sa ograničenjima u objektno-orijentisanom okruženju, odnosno Java programskom jeziku.

2. PROGRAMIRANJE SA OGRANIČENJIMA

Metode programiranja sa ograničenjima (*Constraint Programming* – skraćeno CP) su veoma zastupljene u savremenim aplikacijama za rešavanje realnih problema koje karakteriše tzv. „kombinatorna eksplozija“. Savremene uspešne kompletne industrijske aplikacije bazirane na konceptu programiranja sa ograničenjima omogućene su razvojem tehnologije korišćenja solvera za rešavanje CP problema [3]. U toku razvoja ove tehnologije puno se radilo na razvoju i unapređenju sistema zaključivanja. Između ostalog, rezultat tih napora koji se pokazao kao jedan od uspešnijih, jeste uvođenje globalnih ograničenja (*Global Constraints*). Ukratko, globalna ograničenja su ona koja se primenjuju za sistem kao celinu.

Ova ograničenja nastala su iz potrebe da se prilikom rešavanja realnih, pre svega industrijskih problema, veliki broj jednostavnih ograničenja, koja se često pojavljuju u realnom primeru, zamene jednim opštim ograničenjem. Na taj način se značajno olakšava modeliranje problema. Pri tome se koriste posebne tehnike konzistencije i posebno razvijeni algoritmi prostiranja (propagacije) ograničenja kojima se značajno ubrzava rešavanje problema. Sve to su razlozi koji čine globalna ograničenja interesantnim za teorijska i tehnološka istraživanja kao i za dalje praktične primene.

Ilustrovaćemo primenu samo jednog globalnog ograničenja umesto više tzv. binarnih ograničenja na sledećem veoma poznatom problemu iz oblasti veštačke inteligencije. Koje cifre odgovaraju slovima $\{S,E,N,D,M,O,R,Y\}$ da bi u decimalnom brojnom sistemu važilo *SEND+MORE=MONEY*?

Globalno ograničenje **alldiff(v)**, gde je **v** skup datih slova, odnosno skup cifara od **0** do **9**, zamenjuje pojedinačna ograničenja kojim bi se obezbedilo da jednom slovu odgovara jedna i samo jedna cifra. Globalna ograničenja imaju dve prednosti:

1. dozvoljavaju propagaciju ograničenja koja je bolja nego kod skupa primitivnih ograničenja, i
2. pojednostavljuju proces modeliranja problema dozvoljavajući apstrakcije visokog nivoa.

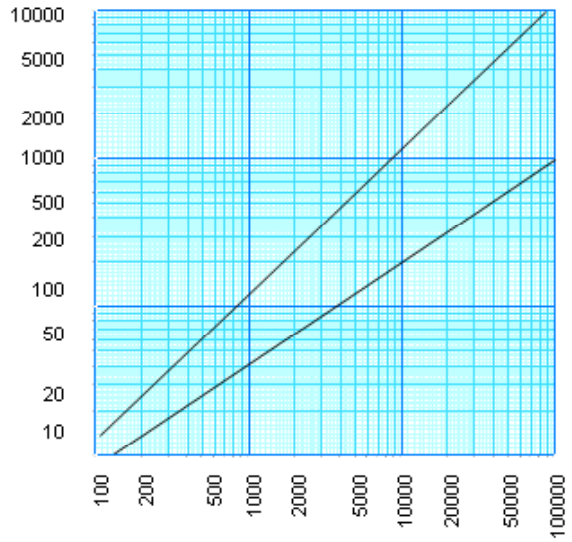
Međutim, ove dodatne funkcionalnosti često znače i dodatnu kompleksnost. Iako su od samog početka CP metode, u poređenju sa drugim metodama, pokazivale dobre rezultate, prošlo je određeno vreme pre nego su se pojavile prve kompletne industrijske aplikacije bazirane na programiranju sa ograničenjima [3].

Razloge, između ostalog, treba tražiti i u potrebi da se razvije okvir koji krajnjim korisnicima omogućava jednostavno korišćenje ograničenja, ali i potreba da tehnologija korišćenja solvera ograničenja dostigne zadovoljavajući nivo. U tom smislu, od posebnog značaja za primenu globalnih ograničenja, važna je mogućnost njihove primene u savremenim objektno-orijentisanim programskim jezicima.

U ovom radu razmatra se praktična primena globalnih ograničenja u Java programskom okruženju, na primeru jednog preduzeća u fazi planiranja resursa potrebnih za pripremu proizvodnje i realizaciju redovnog održavanja mašina.

3. PLANIRANJE ODRŽAVANJA UZ POMOĆ OGRANIČENJA

Planiranje se može smatrati pripremom efikasne realizacije projekta održavanja. U slučajevima preventivnog i investicionog održavanja, kada se radi obavljanja poslova održavanja oprema isključuje iz upotrebe, dobrim planiranjem se može skratiti vreme zastoja uz racionalno korišćenje raspoloživih resursa. Slika 1. pokazuje opseg vremena potrebnog da se pripremi prekid [4].



Slika 1. *Odnos potrebnog vremena za planiranje projekta i broja zahtevanih sati*

Proces planiranja održavanja prati veliki broj ograničenja: trajanje aktivnosti, dostupnost resursa, deljenje resursa, relacije prethođenja itd. Ova ograničenja definišu prostor dozvoljivih rešenja. Često je, u cilju nalaženja bilo kakvog rešenja, problem potrebno dodatno „relaksirati“ dodatnim, prioritetnim ograničenjima. Ova ograničenja često su međusobno konfliktna. U praksi, prioritetna ograničenja se kombinuju u jedinstvenu funkciju koju treba minimizirati, odnosno maksimizirati. Alternativa je kreiranje odgovarajućih heuristika. U oba slučaja, problem raspoređivanja definiše se kao skup ograničenja koja je potrebno zadovoljiti [5].

3.1 Identifikovanje aktivnosti

Na početku planiranja treba identifikovati sve poslove koje treba obaviti u okviru projekta održavanja. Treba razmotriti mogućnost da se u prostoru održavanja za vreme prekida obave i drugi poslovi koji nisu u neposrednoj vezi sa održavanjem opreme [4].

Priprema procesa održavanja može podrazumevati identifikovanje svih aktivnosti koje su na neki način povezane sa održavanjem. Među tim aktivnostima postoji razlika u pogledu efekata kojim utiču na ukupan proces održavanja. Uticaj nekih aktivnosti bi se mogao smatrati značajnim, pa čak i ključnim, ukoliko bi efekti realizacije imali uticaja na ukupan rezultat projekta održavanja. Sa druge strane, pojedine aktivnosti mogu značiti mnogo utrošenog vremena sa malim postignutim efektima.

3.2 Identifikovanje ograničenja

Generalno, broj aktivnosti u projektima varira od nekoliko do više hiljada. To važi i za projekte održavanja. Pojedini algoritimi koji daju dobra rešenja za manje probleme mogu biti potpuno neupotrebljivi kod većih problema. U realizaciju svakog projekta uključen je veliki broj učesnika, različitih profila, pa su razvijene tehnike koje omogućavaju predstavljanje projekata na način razumljiv svima koji su uključeni. Među ovim metodama posebno se ističu metode mrežnog planiranja CPM i PERT, pomoću kojih se planira tok realizacije određenog projekta. Metode mrežnog planiranja se baziraju na grafičkom prikazu redosleda aktivnosti u okviru jednog projekta i njihovih međusobnih zavisnosti. Ovakva logička struktura omogućava detaljnu analizu vremena realizacije pojedinih aktivnosti i projekta u celini. [5].

Aktivnost je najmanja jedinica u okviru projekta koja zahteva vreme i resurse i koja se mora planirati i rasporediti u okviru celog projekta. Odnos prethođenja (*precedence*) definiše redosled izvršavanja pojedinih aktivnosti. Ovaj odnos definiše početak aktivnosti, tako da ona ne može da počne sve dok se sve aktivnosti koje mu prethode ne završe [6].

Sa aspekta primene programiranja sa ograničenjima, u pripremi plana projekta održavanja, može se reći da su važne sve identifikovane aktivnosti. Međutim, problem može da predstavlja utvrđivanje logičkih međuzavisnosti, naročito kod npr. velikih postrojenja. Potrebno je, na adekvatan način, odrediti metode za identifikaciju logičkih zavisnosti.

Generalno, veliki broj kombinatornih problema ima samo jedan uslov – pronalaženje bilo kog rešenja koje zadovoljava sva ograničenja. To se često naziva problemom traženja [7]. Ako se traži i zahteva pronalaženje najboljeg rešenja po nekom kriterijumu, onda se radi o problemu optimizacije. Za problem traženja potrebno je utvrditi da li neko rešenje uopšte postoji. Za problem optimizacije potrebno je utvrditi da li postoji rešenje za zadatu vrednost funkcije cilja.

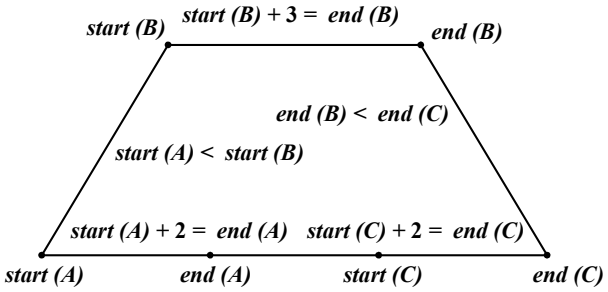
S obzirom na složenost, poželjno je za projekat planiranja održavanja, sa gore navedenim karakteristikama, dodati i deo vezan za prioritete, koji bi olakšali pronalaženje optimalnog rešenja. Zbog toga je predstavljen koncept tvrdih i mekih ograničenja [8,9]. Tvrda ograničenja su ona koja se svakako moraju zadovoljiti prilikom rešavanja problema. Za razliku od njih, broj zadovoljenih mekih ograničenja predstavlja meru kvaliteta dobijenog rešenja. U opštem slučaju, potrebno je minimizarati broj mekih ograničenja koja ne mogu biti zadovoljena. U ovom slučaju, od rasporeda se traži da zadovoljava sva tvrda ograničenja i da minimizira ili maksimizira funkciju cilja koja se odnosi na meka ograničenja.

Odgovarajuće rešenje planiranja projekta održavanja bazirano na programiranju sa ograničenjima, svakako bi imalo mogućnost davanja odgovora na pitanje da li postoji izvodljivo rešenje sa datim ograničenjima. Ako se pokaže da je to potrebno, moguće je razmotriti rasterećenje postojećih ograničenja. Dodavanje novih ograničenja ne bi dovelo do pronalaženja zadovoljavajućeg rešenja jer bi ograničenja koja nisu davala odgovarajuće rešenje i dalje bila razmatrana, a kako CP program zahteva zadovoljenje svih ograničenja javljao bi se isti problem.

Sveobuhvatan proces planiranja resursa podrazumeva planiranje materijala i delova, planiranje radnika i opreme, odnosno planiranje svih pojedinih vrsta resursa neophodnih za realizaciju projekta. Planiranje različitih resursa je veoma složen proces, i ove pojedinačne procese treba sinhronizovati i uklopiti u ukupan proces planiranja resursa, odnosno preko njega u ukupan proces planiranja realizacije projekta [10,11].

3.3 Vremenska ograničenja i ograničenja resursa

Vremenska ograničenja koja važe u jednom projektu često se opisuju odgovarajućim grafom [5]. Primer je prikazan na slici 2.



Slika 2. *Grafički prikaz vremenskih ograničenja*

Uopšteno, ograničenja vezana za raspored aktivnosti mogu se podeliti u dve kategorije: vremenska i ograničenja resursa [1,5]. Vremenska ograničenja, kojima se ujedno opisuje i relacija prethođenja, naječešće se predstavljaju izrazom:

$$A_1 + t \leq A_2$$

gde su A_1 i A_2 promenljive koje predstavljaju vreme početka ili završetka aktivnosti, a t je vremenski period koji mora proteći između ove dve vremenske tačke. Najčešće t predstavlja dužinu trajanja aktivnosti A_1. Ograničenja resursa definišu kapacitet resursa potrebnih za izvršavanje pojedinih aktivnosti.

Generalno, može se reći da se rešavanje problema alokacije resursa odvija u dva smera: od resursa prema aktivnostima, u cilju dobijanja najranijeg (najkasnijeg) početka (završetka) aktivnosti, i od aktivnosti prema resursima, u cilju definisanja ukupnog broja angažovanih resursa.

Prva verzija plana se pravi uz pretpostavku da su raspoloživi resursi dovoljni za njegovu realizaciju. Međutim, može se pokazati da bi planirano paralelno izvođenje nekih aktivnosti i/ili drugih projekata zahtevalo u određenim vremenskim periodima više resursa (radna snaga, oprema, alati, novac...) nego što ih stvarno ima [4].

U skladu s tim, može se u okviru istog plana doći do vremenske neusklađenosti u zahtevima za resursima i u drugom smeru: da se javljaju vremenski periodi u kojima su raspoloživi resursi nedovoljno angažovani što posebno nije dobro ako je u pitanju radna snaga, ili skupa oprema. U takvim slučajevima se moraju uložiti dodatni napori u planiranju radi obezbeđivanja uravnoteženog korišćenja raspoloživih resursa. Ovaj posao u planiranju projekta naziva se nivelacija, balansiranje ili uravnotežavanje korišćenja resursa [4].

Metode programiranja sa ograničenjima mogu da doprinesu ispunjenju zahteva balansiranog korišćenja resursa. To se posebno odnosi na globalna ograničenja.

4. RESURSI I OGRANIČENJA

U realnim uslužnim i proizvodnim sistemima odluke se često donose u uslovima ograničenog broja vrsta i ograničene količine različitih resursa. Ograničena raspoloživost resursa stvara probleme u planiranju aktivnosti, što može da dovede do nepovoljnog raspoređivanja i korišćenja resursa u vremenu.

Resursi se dele u dve kategorije: obnovljivi (*renewable*, *reusable*) i potrošni (*consumable*). Obnovljivi resursi su oni koji se mogu ponovo koristiti i nakon korišćenja u periodu izvršenja neke aktivnosti. Obnovljivi resursi se dele na unarne (*unary*), diskretne (*discrete*) i kontinualni (*continous*). Unarni su oni kojih ima samo u jednom primerku. Na primer, u proizvodnji se često dešava da postoji samo jedna mašina određenog tipa. Diskretni obnovljivi resursi su oni koji se koriste u diskretnim količinama ili komadima. Na primer, radnik ili dizalica ne mogu da se dele. Kontinualni resursi su oni koji se mogu koristiti u, uslovno rečeno, proizvoljnim količinama [12].

Potrošni resursi se troše tokom izvršenja neke aktivnosti. Osim diskretnih i kontinualnih, čije su karakteristike iste kao i kod obnovljivih resursa, postoje još i monotoni i nemonotoni potrošni resursi. Tipični primeri potrošnih resursa su gorivo (kontinualni) ili potrošni materijal za održavanje mašina, na primer zavrtnji (diskretni).

Svaka aktivnost ima svoje trajanje i skup potrebnih resursa. Ovaj skup se može podeliti na nekoliko podskupova nazvanih *grupe resursa* [13, 14]. Svaka od ovih grupa može biti konjuktivna ili disjunktivna. Konjuktivna grupa označava grupu resursa od kojih je svaki potreban za izvršavanje aktivnosti. Disjunktivna grupa znači da je za aktivnost potreban samo jedan od resursa iz grupe.

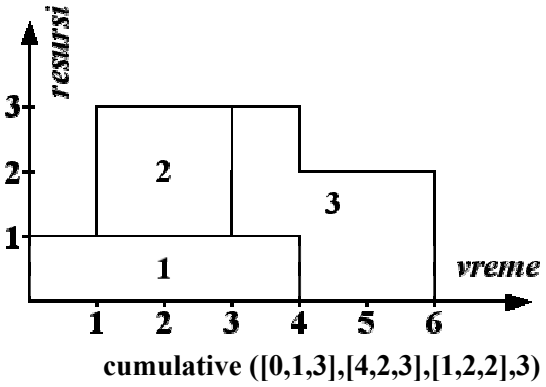
5. NEKA VAŽNA GLOBALNA OGRANIČENJA

Postoji veliki broj razrađenih globalnih ograničenja [15]. Ovde, radi ilustracije, navodimo samo par njih. Ograničenje cumulative se koristi za rešavanje problema alokacije i raspoređivanja. Ograničenje se opisuje na sledeći način [16]:

cumulative([S_1,...,S_n],[D_1,...,D_n], [R_1,...,R_n],L)

gde su **[S1,...,Sn]**, **[D1,...,Dn]** i **[R1,...,Rn]**, neprazni nizovi domena neke promenljive, a **L** neki prirodan broj. Sa **Si** su obeležena vremena početka odgovarajućih aktivnosti, **Di** su trajanja, a **Ri** količina potrebnih resursa za svaku od aktivnosti. **L** predstavlja raspoloživu ukupnu količinu resursa. Ovo ograničenje obezbeđuje da u svakom trenutku izvršavanja aktivnosti, broj angažovanih resursa ne bude veći od zadatog broja **L**.

Na primer, potrebno je rasporediti tri aktivnosti, od kojih prva zahteva jedan od ukupnog broja dostupnih resursa, u trajanju od četiri vremenske jedinice, a aktivnosti 2 i 3 koriste po dve resursne jedinice u trajanju od dva, odnosno tri vremenska perioda. Ako su počeci aktivnosti dati respektivno za svaku aktivnost i iznose 0, 1 i 3, tada se ovo ograničenje može predstaviti kao na slici 3.

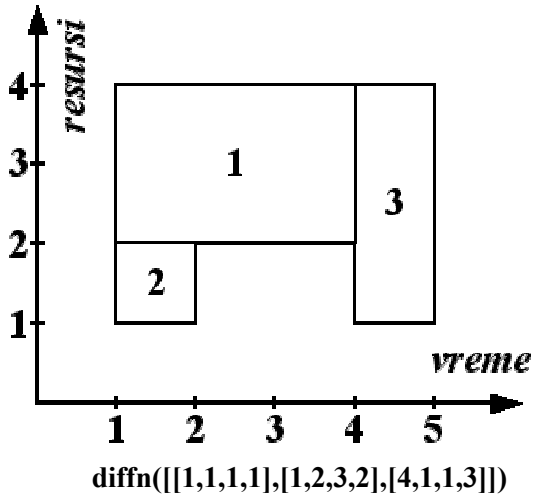


Slika 3. *Primer korišćenja ograničenja* **cumulative**

Ograničenje **diffn** prvi put pojavilo se u CHIP jeziku za rešavanje višedimenzionalnih problema vremenskog rasporeda ili pozicioniranja (*timetabling*) [16]. Kao argumenti ovog ograničenja pojavljuju se n-dimenzionalne figure predstavljene n-torkom $\mathbf{[X_1,...,X_n,L_1,...,L_n]}$ gde je **X** koordinata početka stranice, a **L** dužina stranice figure u dimenziji *i*:

$$\mathbf{diffn([X_1,...,X_n,L_1,...,L_n])}$$

Ograničenje obezbeđuje da se zadate figure ne poklapaju ni u jednom pravcu. U primeru sa slike 2. pravougaonik obeležen brojem 1, predstavljen je četvorkom **[1,2,3,2]**, gde prva dva broja predstavljaju koordinate tačke donjeg levog ugla, a druga dva dužine strana u datim pravcima. Na sličan način pravougaonici 2 i 3 mogu se opisati četvorkama **[1,1,1,1]** i **[4,1,1,3]**, respektivno.



Slika 4. *Primer korišćenja ograničenja* **diffn**

6. GLOBALNA OGRANIČENJA I JAVA

Tehnologija CP programiranja je u svom intenzivnom razvoju dostigla tačku kada je moguće izdvojiti neke osnovne alate i koristiti ih kao dodatke programskih jezika opšte namene, kao što je npr. Prolog. Druga mogućnost je korišćenje posebnih programskih biblioteka koje se koriste u objektno-orijentisanim programskim jezicima.

Popularnost Java programskog jezika dovela je do pojave velikog broja biblioteka koje omogućavaju kombinovanje prednosti ovog okruženja sa prednostima korišćenja metoda CP programiranja. Ovo se posebno odnosi na razvoj sistema za planiranje i raspoređivanje u industriji. U ovom delu navodimo neke od aktuelnih biblioteka.

Cream je biblioteka koja se koristi za razvoj inteligentnih programa koji omogućavaju traženje bilo kakvog (problem zadovoljivosti) ili optimalnog (problem optimizacije) rešenja koje će zadovoljiti skup zadatih ograničenja. *Cream* je u potpunosti razvijen u Java programskom jeziku i predstavlja *Open-source* rešenje [17]. Ograničenja se opisuju na način uobičajen za Java sintaksu. Na raspolaganju su različiti algoritmi za optimizaciju kao što su simulirano kaljenje, tabu pretraživanje itd.

JaCoP biblioteka (*Java Constraint Programming Library*) omogućava definisanje primitivnih ograničenja sa promenljivama čija se rešenja nalaze u konačnom domenu, kao i odgovarajućih metoda pretraživanja. Osim uobičajenih primitivnih ograničenja kao što su jednakosti, nejednakosti, logička i uslovna ograničenja, ova biblioteka omogućava i definisanje globalnih ograničenja: **alldifferent**, **circuit**, **cumulative**, **diff2** i **element**. Uslovi korišćenja ove biblioteke dati su na [18].

Koalog Constraint Solver je Java biblioteka koja omogućava rešavanje i problema zadovoljivosti i problema optimizacije. Osim solvera za rešavanje problema sa standardnim, aritmetičkim i *boolean* ograničenjima, ova biblioteka ima i definisana globalna ograničenja za rešavanje problema raspoređivanja, alokacije resursa, planiranja, rasporeda časova itd. Više informacija o ovoj biblioteci može se naći na [19].

U ovom radu posebno će biti obrađena *Choco* biblioteka, *Open-source* biblioteka koja se distribuira pod uslovima BSD licence. Za sve biblioteke važe relativno slični principi korišćenja koji se mogu podeliti u dve faze:

1. modeliranje problema, i
2. kreiranje i primena solvera koji odgovara datom problemu ili nekoj definisanoj strategiji rešavanja.

Modeliranje problema se obavlja uz pomoć klasa čija imena intuitivno upućuju na njihovu svrhu. Kod pojedinih biblioteka solveri su snažni i dovoljno fleksibilni da bi rešili većinu problema. Neke od biblioteka pružaju korisnicima mogućnost i da samostalno definišu solver, kao što je pomenuta Koalog biblioteka.

7. CHOCO

Choco je Java biblioteka koja se koristi za rešavanje CSP problema (*Constraint Satisfaction Problems*), za praktičnu primenu programiranja sa ograničenjima u Java okruženju i sl. Osnovni element programa napisanog uz pomoć *Choco* biblioteke je objekat klase **Problem** koji se generalno kreira na sledeći način:

```
AbstractProblem prblm =
new Problem();
```

Klasa **Problem** pruža korisnicima API interfejs za dodavanje definisanom problemu promenljivih i ograničenja koji su, kao i odgovarajuće rešenje problema, u vezi sa kreiranim objektom klase **Problem**. Kod *Choco* biblioteke koriste se tri osnovna tipa promenljivih:

1. **IntDomainVar** – promenljive sa mogućim vrednostima u celobrojnom domenu,
2. **RealVar** – promenljive koje imaju kontinualni domen mogućih rešenja, a za predstavljanje vrednosti se koriste intervali, i
3. **SetVar** –za domene diskretnih skupova gde su skupovi vrednosti mogućih rešenja za promenljive.

Kada se kreira problem, promenljive se kreiraju uz pomoć tzv. *factories* metoda, dostupnih za dati problem, a ne uz pomoć klasičnih Java konstruktora. Korišćenje ovih metoda dozvoljava redefinisanje promenljivih u okviru datog problema i osigurava da ograničenja i tipovi promenljivih ostaju kompatibilni.

Za objekat klase **Problem**, promenljiva čije se moguće vrednosti nalaze u konačnom celobrojnom domenu se kreira na sledeći način:

```
IntDomainVar p1 =
prblm.makeEnumIntVar(„prom1“,1,10);
```

gde je **p1** promenljiva čije ime je **prom1**, a moguća rešenja se nalaze u diskretnom domenu sa vrednostima od **1** do **10**. Ova promenljiva se odnosi na već definisan objekat problema – **prblm**.

Za dati problem ograničenje se definiše korišćenjem metoda post, koji je dostupan objektima klase **Problem**:

```
post(Constraint C)
```

Na primer, dodavanje ograničenja koje opisuje relaciju manje ili jednako (≤), između dve promenljive definisanog problema **prblm**, realizuje se na sledeći način:

```
prblm.post(prblm.leq(p1,p2));
```

Kada se problem modelira i kreira, zahvaljujući API interfejsu problem se može rešiti, a dobijeno rešenje prikazati na sledeći način:

```
if (prblm.solve() ==
Boolean.TRUE) {
        for(int i = 0; i
    <prblm.getNbIntVars();
        i++)

    System.out.println(
    prblm.getIntVar(i)+
„=“+((IntDomainVar)
    prblm.getIntVar(i)).
    getVal());
```

Treba uočiti da **solve()** metoda vraća objekat tipa **Boolean**, jer vrednost može biti i **null**.

Klasa **Problem** je centralni element *Choco* modela, budući da omogućava kreiranje i dodavanje promenljivih i ograničenja. Međutim, upravljanje procesom pretraživanja vrši se uz pomoć klase **Solver**. Sledeći kod daje rezultat koji je identičan onom koji se dobija primenom metode **solve()**, uz dodatnu mogućnost da se menjaju parametri.

```
Solver s = prblm.getSolver();
s.setFirstSolution(true);
s.generateSearchSolver(prblm);
// ovde se definišu parametri solvera
s.launch();
Boolean isFeasible = pb.isFeasible();
```

8. KORIŠĆENJE CUMULATIVE OGRANIČENJA

Preduzeće DN company iz Vladičinog Hana bavi se proizvodnjom nameštaja od drveta i želi da postane lider u ovoj oblasti. Proizvodni program je veoma raznovrstan i za njegovu realizaciju koriste se savremene mašine. Budući da je proizvodnja često uslovljena veoma kratkim rokovima, veoma je važno planiranje efikasne pripreme mašina za naredni proizvodni ciklus, uz istovremeno obavljanje svih neophodnih radnji vezanih za održavanje.

Menadžment preduzeća zna da od pravilno organizovanog i ostvarenog procesa održavanja, zavisi i ekonomski rezultat, pre svega zbog smanjenja investicija potrebnih za nabavku nove opreme, sa dužim periodom racionalnijeg iskorišćenja postojećih kapaciteta. Održavanje se, najčešće, sastoji od niza međusobno zavisnih aktivnosti. Osim toga, proces održavanja prati i veliki broj ograničenja, ali i zahtevi da se potrebni resursi za što efikasnije koriste.

Najveći izazov u organizaciji procesa održavanja predstavlja raspoređivanje ograničenih resursa, u ovom slučaju radne snage. Preduzeće DN radi u dve smene, u kojima je angažovano po sedam radnika iste kvalifikacije. Najveći izazov za organizaciju proizvodnje predstavlja čest slučaj kada je potrebno tokom jednog radnog dana, u dve smene, završiti prilagođavanje mašina potrebama novog proizvodnog ciklusa, ali i završiti sve aktivnosti vezane za održavanje.

U tim uslovima, koji obično podrazumevaju vremenska i druga ograničenja, potrebno je postojeće resurse optimalno organizovati. U slučaju preduzeća DN to najčešće znači 30 standardnih aktivnosti, a najveći problemi vezani su za raspoređivanje radne snage. U nastavku je prikazan postupak primene globalnih ograničenja koja preduzeću treba da pomognu prilikom planiranja angažovanja ljudskih resursa.

Ograničenja koja prate pripremu proizvodnje i održavanja, a odnose se na međusobnu zavisnost odvijanja aktivnosti i na potreban broj resursa za svaku aktivnost, obično se definišu tabelarno. Dužina trajanja aktivnosti definisana je brojem potrebnih sati u okviru jedne smene. U tabeli 1. prikazan je deo definisanih aktivnosti.

Tabela 1.

Oznaka	Trajanje (časova)	Broj potrebnih resursa	Prethodne aktivnosti
Aktivnost 1	1	2	-
Aktivnost 2	2	1	1
Aktivnost 3	1	4	istovremeni završetak sa 2
Aktivnost 4	3	2	istovremeni početak sa 3
...	...	...	...

Na početku je definisan odgovarajući problem:

```
Problem plan = new Problem();
```

Za svaku aktivnost definisane su četiri promenljive: pocetak, kraj, trajanje i resursi. Predviđene dužine trajanja aktivnosti i potrebni resursi predstavljaju definisane i konstante vrednosti, i za potrebe ovog rada zadate su kao **int[]** tabele:

```
int[] resursi_data =
new int[]{2, 1, 4, 2...};
int[] trajanje_data =
new int[]{1, 2, 1, 3...};
```

Zatim su u modelu definisana ograničenja. Prvo se dodaje globalno ograničenje **cumulative** koje obezbeđuje da broj angažovanih resursa, u svakom trenutku izvršavanja aktivnosti, odgovara raspoloživom kapacitetu:

```
plan.post(plan.cumulative(
pocetak, kraj,
trajanje, resursi, kapacitet));
```

Realizacija ovog ograničenja u Choco biblioteci, podrazumeva da se kao argumenti koriste promenljive **pocetak** i **kraj**, dok su **trajanje** i **resursi** konstante definisane za svaku aktivnost. Osim toga, pojavljuje se i konstanta **kapacitet** koja u ovom slučaju odgovara broju radnika koji su na raspolaganju u obe smene:

```
int capa = 7;
```

Planom realizacije predviđeno je i da Aktivnost 2 ne počne pre nego što se završi Aktivnost 1:

```
pb.post(pb.leq(pb.plus(starts[1],
duration[1]),starts[2]));
```

Konačno, potrebno je obezbediti i odgovarajući redosled aktivnosti tako da se Aktivnost 2 završava istovremeno kada i Aktivnost 3, odnosno da Aktivnost 3 počinje kada i Aktivnost 4:

```
pb.post(pb.eq(pb.plus(starts[2],
duration[2]), pb.plus(starts[3],
duration[3])));
pb.post(pb.eq(starts[3],
starts[4]));
```

Aplikacija rešava dati problem za 125 milisekundi i za prve četiri aktivnosti rešenje se prikazuje na sledeći način:

```
Aktivnost 1 - pocetak: 0 –
zavrsetak: 1 - radnika: 2
Aktivnost 2 - pocetak: 1 –
zavrsetak: 3 - radnika: 1
Aktivnost 3 - pocetak: 2 –
avrsetak: 3 - radnika: 4
Aktivnost 4 - pocetak: 2 –
zavrsetak: 5 - radnika: 2
...
```

9. ZAKLJUČAK

Rešavanje problema planiranja resursa znatno je olakšano primenom globalnih ograničenja. Mogućnosti primene metoda programiranja sa ograničenjima, a naročito globalnih ograničenja, za rešavanje realnih problema su velike. Ono što ove metode posebno izdvaja je deklarativnost, pa se pisanje programa praktično svodi na opisivanje problema, a uz pomoć odgovarajućih solvera dolazi se do rešenja.

Poslednjih godina razvijen je niz paketa koji omogućavaju primenu ovih metoda u objektno-orijentisanom okruženju. Želja je da se na ovaj način obezbedi mogućnost jasnog opisivanja odnosa između različitih entiteta nekog programa. U ovom radu opisani su osnovni elementi korišćenja globalnih ograničenja za planiranje resursa u Java programskom okruženju.

U teškim uslovima proizvodnje, neizvesna je mogućnost korišćenja svih potrebnih resursa u procesu pripreme proizvodnje i održavanja. U takvim uslovima potrebna je dobra organizacija, pre svega sa aspekta planiranja angažovanja ograničenog broja resursa. Korišćenje globalnih ograničenja pruža velike mogućnosti za rešavanje ovakvih problema, naročito sa aspekta krajnjeg korisnika, koji nije u obavezi da poznaje npr. programiranje. U ovom radu prikazani su osnovni elementi korišćenja globalnih ograničenja realizovanih u Java programskom za planiranje resursa u jednom preduzeću.

LITERATURA

[1] Maynard, H. B.: *Savremena organizacija proizvodnje*, Gornji Milanovac, RO Kulturni centar, JUR Privredna knjiga, 1980.

[2] Choueiry, B. Y., Faltings, B., Noubir, G.: *Abstraction Methods for Resource Allocation*, Technical Report No. TR-94/47, Département d'Informatique, Ecole Polytechnique Fédérale de Lausanne, Switzerland, 1994.

[3] Simonis H.: *Models for Global Constraint Applications*, Constraints, Springer Science, 2006.

[4] Vujošević, M., Upravljanje projektima u održavanju, *Istraživanja i projektovanja za privredu*, Vol. IV, (2006), No 13, str. 52-64

[5] Le Pape, C., Implementation of Resource Constraints in Ilog Schedule: A Library for the Development of Constraint-Based Scheduling Systems, *Intelligent Systems Engineering*, Vol. 3 (2), 1994, pp. 55-66

[6] Krajewski, L. J., Ritzman, L. P.: *Operations management – Strategy and Analysis*, Fourth Edition, Addison-Wesley Publishing Company, 1996.

[7] Schaerf, A.: *A survey of automated timetabling*, Technical Report CS-R9567, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, The Netherlands, citeseer.nj.nec.com/context/2080028/0, 1995.

[8] Bistarelli, S., Codognet, P., Georget, Y., Rossi, F.: *Abstracting Soft Constraints*, New Trends in Constraints, 1999, pp. 108-133

[9] Rossi, F.: *Constraint Logic Programming*, http://citeseer.ist.psu.edu/422446.html

[10] Jovanović, P.: *Upravljanje projektom – Project Management*, 4. izdanje, Beograd, Grafoslog, 1999.

[11] Lever, J., Wallace, M., Richards, B.: *Constraint logic programming for scheduling and planning*, British Telecom Technology Journal, Vol.13, No.1., 1995, pp. 73-80

[12] Haslum, P., Geffner, H.: *Heuristic Planning with Time and Resources*, http://citeseer.nj.nec.com/haslum01heuristic.html

[13] Baptiste, P., Le Pape, C.: *Disjunctive Constraints for Manufacturing Scheduling: Principles and Extensions*, Proceedings of the Third International Conference on Computer Integrated Manufacturing, Singapore, 1995.

[14] Müller, T., Bartak, R.: *Interactive timetabling*, Proceedings of ERCIM Workshop on Constraints, Prague, 2001.

[15] Beldiceanu N., Carlsson M., Rampon J.X.: *Global Constraint Catalog*, SICS Technical Report T2005:08, Swedish Institute of Computer Science, Sweden, 2005.

[16] Goltz H.J., Küchler G., Matzke D.: *Constraint-based Timetabling for Universities*, Proceednigs of 11th International Conference on Applications of Prolog, Tokyo, 1998.

[17] http://bach.istc.kobe-u.ac.jp/cream/

[18] http://www.cs.lth.se/home/Radoslaw_Szymanek/

[19] http://www.koalog.com/php/jcs.php