VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
MATEMATINĖS STATISTIKOS KATEDRA

AKCIJŲ PORTFELIO MODELIAVIMAS

Jurgita Gilytė

Magistro darbas

Darbo vadovas
doc. P. Vaitkus

Vilnius
2006

TURINYS

1 ĮVADAS

Portfelio parinkimo uždavinys yra viliojantis technikos moksluose, tiesioginiuose („online“) algoritmuose ir žinoma financiniuose skaičiavimuose. Internetinėje svetainėje www.google.com surinkus užklausą „portfolio optimization“, gauname 381000 rezultatų. Taigi straipsnių šia tema yra tūkstančiai. Pagrindinis modelis yra β, kai maksimizuoja vidurkį, minimizuodami dispersiją. Naujasis požiūris yra vaR (value at Risk) – nuostolių tikimybių apribojimas.

Šiame darbe bandoma efektyviau pritaikyti *ANTICOR* algoritmą naudojant stabdymo funkciją ir genetinį algoritmą. Skaičiavimams naudojami istoriniai duomenys, nes naujų rasti nebegalima. Lietuvos duomenys dar nenagrinėti. Tai nėra akcijų kitimo prognozė. Kalbant apie Lietuvą, mes negalėtume numatyti įmonių UAB „Ekranas“ ir UAB „Vingis“ bankroto.

2 PORTFELIO PARINKIMO PROBLEMA

Tarkime, kad biržoje yra m akcijų. Tegu $v_t = (v_t(1),...,v_t(m))$ yra m akcijų kainų vektorius, uždarant biržą laiko momentu t. Čia laiko momentas yra para, o $v_t(j)$ - j-os akcijos kaina. Taip yra patogiau dirbant su santykinėmis kainomis $x_t(j) = v_t(j)/v_{t-1}(j)$, todėl investuojant d dolerių į j-ąją akciją prieš laiko momentą t, mes turime $dx_t(j)$ dolerių. Tarkime, kad $x_t = (x_t(1),...,x_t(m))$ yra santykinių kainų biržos vektorius laiko momentu t. Portfelis b yra lėšų paskirstymas kiekvienai akcijai, susidentis iš turimo dolerio vertės, investuotos į kiekvieną akciją santykių $b = (b(1),...,b(m))$. Čia $b(j) \geq 0$ ir $\sum_j b(j) = 1$. Portfelio b ir biržos vektoriaus x sandauga - paros duodama grąža $b \cdot x = \sum_j b(j)x(j)$. Visa grąža žymima $ret_X(b_1,...,b_n)$ ir yra lygi $\prod_{t=1}^{n} b_t \cdot x_t$, čia $b_1,...,b_n$ - portfelių seka, o $X = x_1,...,x_n$ - biržų seka. Portfelio parinkimo algoritmas A yra bet kuri apibrėžtumo arba atsitiktinumo taisyklė, apibrėžianti portfelių seką. Tegu $ret_X(A)$ pažymi biržų sekos X visą gaunamą grąžą.

3 ANTICOR ALGORITMAS

ANTICOR algoritmas nebando atspėti akcijų lyderio, o yra paremtas konstantų pertvarkymo algoritmu ir blogiausio rezultato radimo universalioje prekyboje loginiu pagrindu. Borodino (Borodin A.), Eljanivo (El-Yaniv R.) ir Gogano (Gogan V.) [1] pasiūlytas algoritmas ne tik iš esmės pasiekia geriausią rezultatą istorinėse biržose, bet ir randa geriausią akciją.

Esamai dienai t, imame w paskutiniųjų prekybos akcijomis dienų, kur w yra sveikas skaičius. Kiekvienos i akcijos tam tikro lango metu augimo tempas yra lygus santykinei kainai tuo metu. Darome prielaidą, kad turime akcijų, kuriomis yra prekiaujama vienodai ilgai, portfelio augimo tempus. ANTICOR algoritmo pirmoji pinigų perkėlimo iš akcijos i į akciją j sąlyga yra ta, kad akcijos i augimo tempas turi viršyti akcijos j augimo tepą pasirinkto laiko lango atveju. Papildomai, ANTICOR algoritmas reikalauja, kad akcija j pradėtų mėgdžioti paskutinį akcijos i augimą artimoje ateityje. Algoritmas reikalauja teigiamos koreliacijos tarp akcijos i priešpaskutinio lango metu ir tarp akcijos j paskutinio lango metu.

Santykinis dydis į kurį mes perkelsime pinigus iš akcijos i į akciją j priklausys nuo to kokio styprumo yra ši koreliacija ir sava anti-koreliacija tarp akcijų i ir j (dviejuose vienas paskui kitą einančiuose languose). Algoritmas yra pavadintas ANTICOR, nes naudojamos koreliacijos ir antikoreliacijos nuosekliuose languose norint rasti akcijų i ir j antikoreliacinę augimo tempo galimybę artimoje ateityje. Tikimasi, kad akcijos j augimo tempas yra didesnis nei akcijos i.

Pažymime,

$$LX_1 = \log(x_{t-2w+1}),...,\log(x_{t-w})^T$$

$$LX_2 = \log(x_{t-w+1}),...,\log(x_t)^T,$$

čia $\log(x_t)$ yra lygus $(\log(x_k(1)),...,\log(x_k(m)))$. LX_1 ir LX_2 yra dviejų vektorių sekos (atitinkamai, dvi $w \times m$ matricos) gautos imant biržos sekos logaritmą atitinkamiems langams $[t-2w+1, t-w]$ ir $[t-w+1, t]$ šia tvarka. LX_k yra j-asis stulpelis, gautas iš $LX_k(j)$, o $\mu_k = (\mu_k(1),\ldots,\mu_k(m))$ yra vektorius, sudarytas iš LX_k stulpelių vidurkių. Panašiai, σ_k yra LX_k stulpelių standartinių nuokrypių vektorius. Kryžminės koreliacijos matrica (ir jos normalizacija) tarp LX_1 ir LX_2 stulplelių vektorių yra žymima taip:

$$M_{\text{cov}}(i,j) = (LX_1(i) - \mu_1(i))^T (LX_2(j) - \mu_2(j));$$

$$M_{cor}(i,j) = \begin{cases} \dfrac{M_{\text{cov}}(i,j)}{(w-1)\sigma_1(i)\sigma_2(j)}, \text{ kai } \sigma_1(i), \sigma_2(j) \neq 0; \\ 0, \text{ kitu atveju.} \end{cases}$$

$M_{cor}(i,j) \in [-1,1]$ yra lygi akcijos i iš pirmo lango ir akcijos j iš antro lango log-santykinių kainų koreliacijai.

Jeigu $\sigma_1(i)$ (atitinkamai $\sigma_2(j)$) kažkuriame lange yra lygi 0, tada akcijos i augimo tempas paskutinio antro lango metu (atitinkamai, akcijos j paskutinio pirmo lango metu) yra konstanta šiame lange. Paėmus pakankamai didelį langą, kiekvienos akcijos i konstantos augimas nieko gero nežada. Tačiau, šiuo nemaloniu atveju yra pasirenkamas būdas, kai su akcija i nieko nedaroma, t.y. nei investuojama, nei neinvestojama. Kiekvienai akcijų i ir j porai skaičiuojamas $claim_{i \to j}$, dydis į kurį norime perkelti savo investicijas iš akcijos i į akciją j. Būtent, jei $\mu_2(i) > \mu_2(j)$ ir $M_{cor}(i,j) > 0$, tada

$$claim_{i \to j} = \alpha M_{cor}(i,j) + \beta A(i) + \gamma A(j), \qquad (1)$$

čia $\alpha, \beta, \gamma = 1$ ir $A(h) = |M_{cor}(h,h)|$, jei $M_{cor}(h,h) < 0$, kitu atveju 0. $M_{cor}(i,j) > 0$ yra naudojamas norint nuspėti ar akcijos i ir j yra koreliuojamos einančiuose iš eilės languose ir

$M_{cor}(h,h) < 0$ numato, kad akcija h bus neigiamai autokoreliuota nuosekliuose languose. Galiausiai,

$$transfer_{i\to j} = b_t(i) \cdot claim_{i\to j} \Big/ \sum_j claim_{i\to j}$$

ir

$$b_{t+1}(i) = b_t(i) + \sum_{j\neq i} \left[transfer_{j\to i} - transfer_{i\to j} \right].$$

ANTICOR algoritmas naudoja lango dydį w, esamą laiko momentą t, istorinės biržos seką X_t ir dabartinį akcijų portfelį $\hat{b}_t$, kuris yra apibrėžtas taip

$$\hat{b}_t = \frac{1}{b_t x_t}\left(b_t(1)x_t(1),...,b_t(m)x_t(m)\right).$$

Algoritmas iš pradžių naudoja tuščius istorinės biržos rezultatus ir pastovų portfelį $\hat{b}_t$ (sudarytą iš m akcijų). Algoritmas gražina naują portfelį, pagal kurį mes turime perskirstyti savo turimas akcijas laiko momentui $(t+1)$, t.y. kitos dienos pradžiai.

4 GENETINIAI ALGORITMAI

Genetiniai algoritmai yra vienas iš evoliucinio skaičiavimo metodų. Apskritai evoliuciniai skaičiavimo metodai iš kitų išsiskiria tuo, kad pagrindiniai jų konstrukcijos ir realizacijos elementai remiasi gerai žinomomis evoliucijos savybėmis. Paprastai tariant, problemos šiuo metodu yra sprendžiamos evoliuciniu procesu, kurio metu gaunamas geriausias sprendinys. Žinomos šios evoliucinių skaičiavimo metodų klasės: genetiniai algoritmai, evoliucinis programavimas, evoliucinės strategijos bei klasifikavimo sistemos. Šiame darbe bus taikomi tik genetiniai algoritmai, todėl kiti evoliucinio skaičiavimo metodų nagrinėjami nebus.

Genetinius algoritmus atrado ir išvystė J. Holandas (J. Holland) [2]. J.Holandas studijavo natūralios adaptacijos fenomeną ir ieškojo būdų kaip jį perkelti į kompiuterį. Genetiniai algoritmai buvo pristatyti kaip biologinės evoliucijos modeliavimo priemonė.

Genetiniai algoritmai, jų veikimas yra pagrįstas evoliucijos, vykstančios gyvojoje gamtoje, t. y., natūraliosios atrankos proceso imitavimu. Pagrindinės sąvokos, kurios naudojamos modeliuojant biologinės evoliucijos procesus, yra *individas* ir *populiacija*. Individas yra tam tikras elementarus, daugiau neskaidomas vienetas, objektas. Didesnė ar mažesnė individų grupė sudaro *populiaciją*. Dar vienas svarbus dalykas yra vadinamasis *individo tinkamumas* − savotiška *individo vertė*, t.y. individo sugebėjimas sėkmingai prisitaikyti (prie aplinkos) ir

išlikti. Šia prasme vertingesnis (grynai biologiškai) yra tas individas, kuris sugeba geriausiai prisitaikyti ir, gal būt, palikti didesnį palikuonių (vaikų) skaičių.

Optimizavime vietoje sąvokų individas, populiacija, individo vertė naudojamos tradicinės, įprastos sąvokos: individui atitinka atskiras sprendinys, populiacijai − sprendinių aibė (grupė, rinkinys), pagaliau, individo vertė yra asocijuojama su tikslo funkcijos reikšme duotajam sprendiniui. Taip, kaip gyvosios gamtos evoliucijoje išlieka tik vertingiausi (stipriausi) individai, taip ir optimizavime siekiama gauti kuo geresnį sprendinį, t. y., sprendinį su kuo mažesne (ar didesne) − nelygu, koks optimizavimo uždavinys (minimizavimo ar maksimizavimo) sprendžiamas − tikslo funkcijos reikšme.

Tradiciškai genetinių algoritmų genomams aprašyti naudojamos dvejetainės išraiškos. Tačiau plečiantis uždavinių spektrui ilgainiui buvo pradėtos naudoti ir kitokios genomo išraiškos: sveikaskaitinės, realieji skaičiai, vektoriai, masyvai, grafai (neuroniniai tinklai).

Klasikinė genetinio algoritmo schema pateikta 1 pav. Pirmame pateiktos schemos žingsnyje (*Pradžia*) sukuriama pradinė sprendinių aibė (populiacija). Paprasčiausias pradinių sprendinių parinkimo būdas yra jų generavimas atsitiktine tvarka. Populiacijos dydis nesikeičia algoritmo vykdymo metu, taigi jis yra vienas iš genetinio algoritmo parametrų.

Įvestis: tikslo funkcija f.
Išvestis: geriausias rastas sprendinys.
1. **[Pradžia]** generuoti pradinių sprendinių populiaciją, kurios dydis yra n.
2. **[Tinkamumas]** apskaičiuoti kiekvieno sprendinio tinkamumą pagal tikslo funkciją $f(s)$.
3. **[Nauja populiacija]** sukurti naują populiaciją kartojant sekančius žingsnius tol, kol populiacija prisipildys.
3.1 **[Išrinkimas]** Pagal tinkamumą išrinkti du tėvinius sprendinius iš populiacijos (kuo didesnis tinkamumas, tuo sprendinys turi didesnę tikimybę būti išrinktas).
3.2 **[Kryžminimas]** Iš tėvynių sprendinių sukuriamas naujas sprendinys (vaikas).
3.3 **[Mutacija]** Pagal mutacijos tikimybę keisti sprendinį (vaiką) atskirose jo dalyse.
3.4 **[Priėmimas]** Įdėti sprendinį (vaiką) į naują populiaciją.
4. **[Pakeitimas]** Tolimesniame darbe naudoti naujai sugeneruotą populiaciją.
5. **[Tikrinti]** Jei pabaigos sąlyga tenkinama, sustabdyti algoritmą ir grąžinti geriausią sprendinį iš einamos populiacijos.
6. **[Ciklas]** Grįžti į žingsnį 2.

1 pav. *Genetinio algoritmo schema.*

Toliau algoritmas vykdo ciklą, kuris gali būti apribotas maksimaliu iteracijų skaičiumi arba vykdymo laiku. Kiekvienoje ciklo iteracijoje stengiamasi pereiti prie geresnės populiacijos. Visų pirma įvertinama esama populiacija (*Tinkamumas*). Toliau kuriama nauja (*Nauja populiacija*). Naują populiaciją kuriame iš senosios pritaikius genetinius operatorius: kryžminimą (*Išrinkimas*, K*ryžminimas*) ir mutaciją (*Mutacija*).

5 TYRIMO DALIS: ANTICOR ALGORITMO TOBULINIMAS

5.1 Algoritmo koeficientų parinkimas

Kiek investicijų iš akcijos i turime perkelti į akciją j, naudojant ANTICOR algoritmą, parodo dydis $claim_{i\to j}$, kuris tiesiogiai priklauso nuo koreliacijų tarp akcijų ir yra apibrėžtas formule (1). ANTICOR algoritme naudojami koeficientai α, β, γ yra lygūs 1. Tačiau ar tikrai šie koeficientai yra optimaliausi ir gautas rezultatas yra geriausias?

Koeficientų radimui naudojame genetinį algoritmą su tikslo funkcija pateikta 2 pav. ir iteracijų skaičiumi lygiu 10.

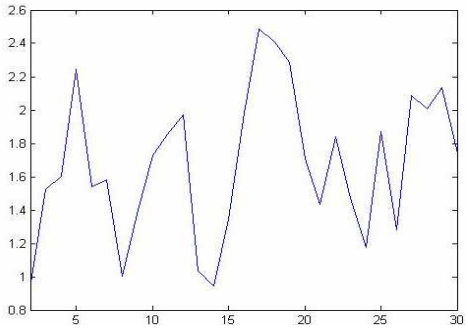
```
function fit = anticor_fitness(g_coef,data,window,t,portfolio);
  if (g_coef(1) < 0 | g_coef(2) < 0 | g_coef(3) < 0)
       fit = 100000000;
  else
      portfolio1 = anticor(data,window,t,portfolio,g_coef);
      ret = data(t+1,:)*portfolio1';
      fit = 1/ret;
  end
```

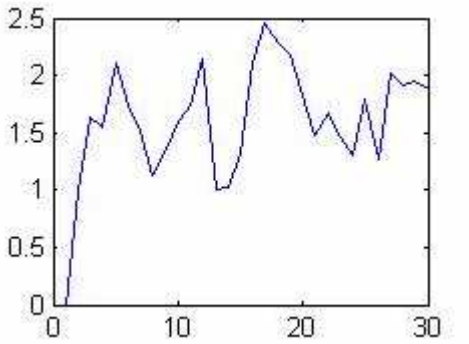
2 pav. *Tikslo funkcija koeficientų radimui.*

Šis algoritmas turi kritinį parametrą – lango dydį w. 3 pav. pavaizduotuose grafikuose matome pelno priklausomybę nuo lango dydžio $w = 2,...,30$. Grafike (a) yra pavaizduotas pelnas, gautas naudojant ANTICOR algoritmą su koeficientais, rastais genetinio algoritmo pagalba. Kaip ir ANTICOR algoritmo kūrėjų grafike (b) didelių pelno šuolių išvengti nepavyksta.

Lygindami (a) ir (b) grafikus (pav.3), matome, kad nėra didelių pelno skirtumo amplitudžių. Algoritmo vaizdavimas visiškai priklauso nuo pasirinkto lango dydžio. Todėl 4 pav., atsitiktinai parinkus kelis lango dydžius, matome, kaip pelnas keičiasi investavimo

(a) (b)

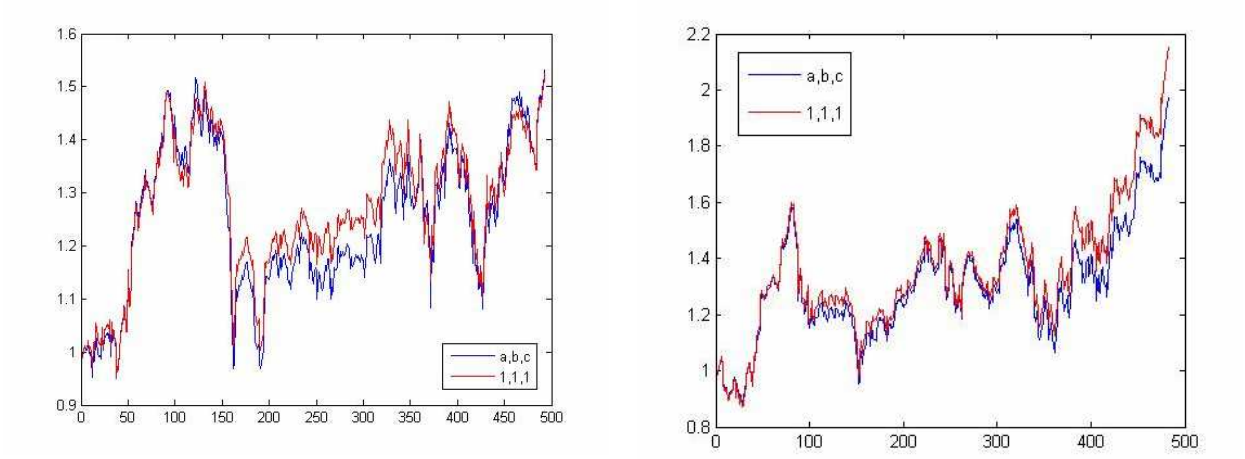




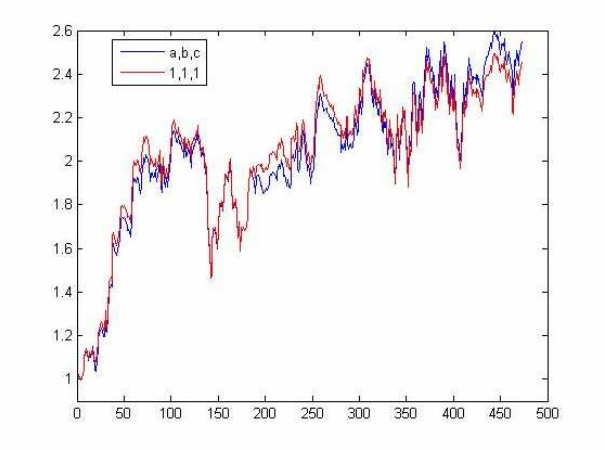
3 pav. *$ANTICOR_w$ algoritmu gauto pelno (investavus 1 dolerį) priklausomybė nuo lango dydžio* $2 \le w \le 30$, *naudojant DJIA duomenis: (a) koeficientų* α, β, γ *radimui naudojamas genetinis algoritmas; (b)* $\alpha = \beta = \gamma = 1$.

laikotarpiu. Atidžiau pažvelgus į grafiką (a), matome, kad rezultatai, pasirinkus investavimo laikotarpį iki 150 dienų yra praktiškai tokie patys kaip ir gauti naudojant algoritmo kūrėjų pasiūlytus koeficientus. Pasirinkus 150 – 350 dienų laikotarpį ryškiai matome, jog mūsų nagrinėjamas algoritmas (žr. priedas Nr. 1) akcijų portfelius parinko nesėkmingai. Tačiau matome, jog nuo 350 dienų iki laikotarpio pabaigos abiejų algoritmų gauti rezultatai supanašėja. Grafikas (b) mums parodo, jog praktiškai visą investavimo laikotarpį, o ypač laikotarpio pabaigoje naudotas genetinis algoritmas yra nesėkmingas. Grafike (c) naudoto algoritmo pelnas yra mažesnis, nors ir pačioje investavimo laikotarpio pabaigoje galime įžvelgti geresnius rezultatus. Apibendrinus šiuos tris grafikus, matome, jog naudojamo genetinio algoritmo koeficientų parinkimas atnešė geresnius rezultatus kai kuriais laiko momentais, nei kad buvo iki šiol.

(a) (b)



(c)

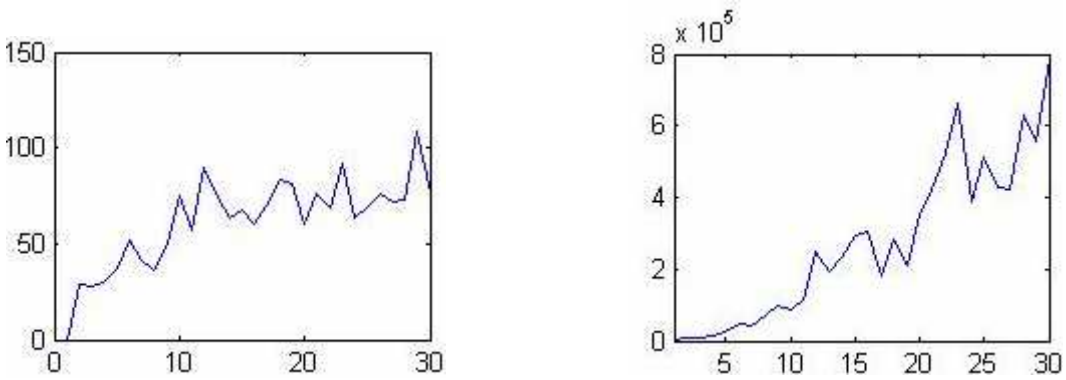


4 pav. *$ANTICOR_w$ algoritmu, kai koeficientų radimui naudojamas genetinis algoritmas, gauto pelno (investavus 1 dolerį) priklausomybė nuo dienų skaičiaus su DJIA duomenimis: (a) w=7; (b) w=12; (c) w=17.*

5.2 Algoritmo stabdymas

Algoritmo stabdymas nenaudoja jokio papildomo algoritmo ir remiasi tuo, kad jei akcijų portfelio pagalba laiko momentu t investavome pinigus ir laiko momentu $t+1$ gavome grąžą mažesnę už 1, tai savo akcijų portfelio nekeičiame ir paliekame iki kitos dienos, kol gauta grąža bus daugiau nei 1. Bet, kaip ir anksčiau minėti algoritmai, taip ir šis turi kritinį parametrą – lango dydį w. 5 pav. pavaizduotas pelnas, rastas taikant ANTICOR algoritmo stabdymą, pasirinkus funkciją, kuri priklauso nuo lango dydžio $w = 2,...,30$.

(a) (b)



5 pav. *$ANTICOR_w$ algoritmu gauto pelno (investavus 1 dolerį) priklausomybė nuo lango dydžio $2 \le w \le 30$, kai naudojama algoritmo sustabdymo funkcija: (a)DJIA; (b) SP500.*

Algoritmo (žr. priedą Nr. 2) gauti rezultatai yra žymiai geresni, nei matėme skyrelyje 5.1. Tačiau algoritmo vaizdavimas priklauso nuo lango dydžio. Tai nėra praktiška, nes aktyvoje biržoje lango dydžio nepasirinksi, todėl kitame skyrelyje bandysime optimizuoti lango dydį w.

5.3 Optimalaus lango radimas

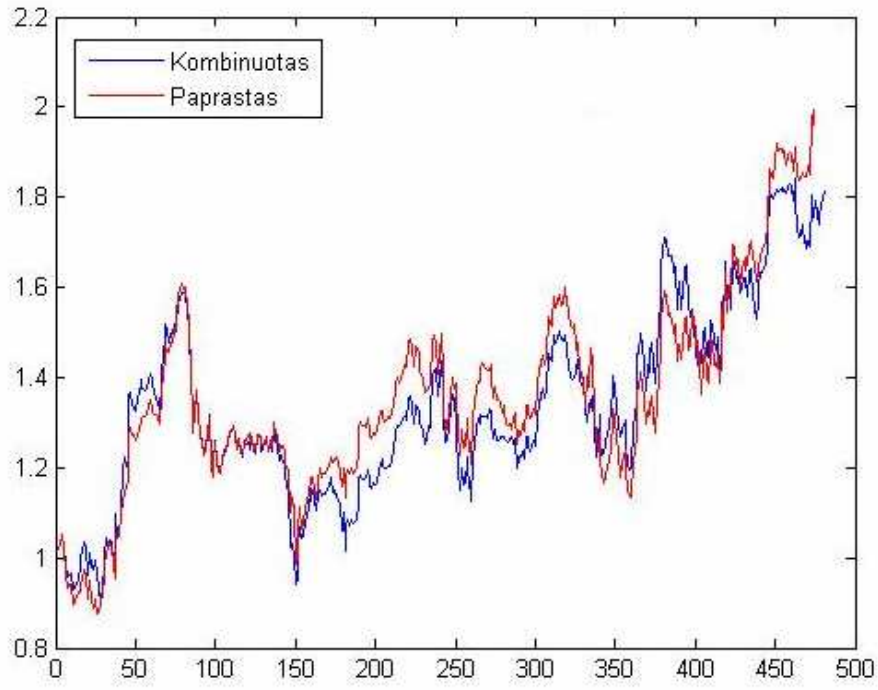
Didžiausia ANTICOR algoritmo problema – lango dydis w. Parametras w parodo kokios apimties praeitį mes analizuojame norėdami parinkti tinkamą akcijų portfelį. Paėmę du vienas paskui kitą einančius langus, bandome juos optimizuoti genetinio algoritmo pagalba. Optimizavimui naudojame tikslo funkciją pavaizduotą 6 pav.

```
function fit = anticor_fitness(g_coef,data,windows,t,portfolio);
  if (sum(g_coef < 0) > 0 | sum(g_coef > 1) > 0)
       fit = Inf;
  else
      portfolio1 = g_coef(1)*anticor(data,windows(1),t,portfolio) +
  (1 - g_coef(1))*anticor(data,windows(2),t,portfolio);

      ret = data(t+1,:)*portfolio1';
      fit = 1/ret;
  end
```

6 pav. *Tikslo funkcija optimalaus lango radimui.*

7 pav. pavaizduotame grafike matome, kaip priklauso gaunamas pelnas nuo investuojamų dienų skaičiaus. Čia „kombinuotas“ grafikas yra gautas, naudojant genetinį algoritmą su itercijų skaičiumi lygiu 50. Genetinis algoritmas ieško geriausio rezultato tarp dviejų langų dydžių $w = 12$ ir $w = 13$, t.y. ieško koeficiento, kuris parodytų tam tikro lango vertę. Atidžiau pažvelgus į grafiką matyti, kad investavimo laikotarpio pradžioje “kombinuotas” algoritmas (žr. priedą Nr. 3) davė geresnį rezultatą, nei „paprasto“ algoritmo metu, kai yra imamas vienas langas $w = 13$. Pasirinkus 70 – 130 dienų investavimo laikotarpį gautas rezultatas yra panašus, o pasirinkus ilgesnį nei 130 ir trumpesnį nei 330 dienų laiko tarpą, „kombinuoto“ algoritmo pagalba gautas pelnas yra ryškiai mažesnis. Investavimo laikotarpiu nuo 330 iki 425 dienų matome teigiamus algoritmo rezultatus, tačiau nuo 425 dienos iki pačios investavimo laikotarpio pabaigos pastebime, kad „kombinuoto“ algoritmo rastas pelnas yra žymiai mažesnis, nei tas kurį gavome naudodami „paprastą“ algoritmą.



7 pav. *Pelno priklausomybė nuo laiko momento t, naudojant DJIA biržos duomenis.*

6 ALGORITMŲ REZULTATAI

Palyginkime algoritmų duodamą pelną. Lyginimui naudojame dvi istorines akcijų biržas. Pirmoji akcijų birža susideda iš 25 akcijų ir yra paimta iš SP500 (Standard & Poor’s 500). Ši akcijų birža apima 1276 dienų laikotarpį nuo 1998 sausio 2 iki 2003 sausio 31. Antroji akcijų birža susideda iš 30 akcijų priklausančių Dow Jones pramonės vidurkiui (DJIA) ir apima

dviejų metų laikotarpį (507 dienas) nuo 2001 sausio 14 iki 2003 sausio 14. Šios akcijų biržos yra gana skirtingos. Norint maksimizuoti gaunamą naudą iš šių duomenų bazių, mes perbėgame kiekvieną biržą ją apversdami. Apverčiame eiliškumą ir investuojame į santykines kainas. Apverstosios duombazės yra žymimos su minuso ženklu „-1“ laipsnyje.

Lentelė Nr. 1 *Pelnas, gautas investavus 1 dolerį, įvairių algorimų pagalba, naudojant dviejų biržų duomenis ir atvirkštinius jų duomenis.*

Algoritmas	SP500	DJIA	$SP500^{-1}$	$DJIA^{-1}$
$ANTI^{1}$	5,2053	1,6735	6,2052	3,7437
Stopping	1043172,0753	127,8099	267554,1253	107,0497
Coef_opt	4,9146	1,6282		3,7553
Window_opt		1,2117		

Lentelėje Nr. 1 gauti rezultatai yra įspūdingi, tačiau realybėje tokių skaičių gauti nepavyktų, nes yra keli periodai, kai sukauptas pelnas šių strategijų metu mažėja (4, 7 pav.). Šis elgesys parodo, kad yra tam tikras rizikos laipsnis naudojant šiuos algoritmus.

Lentelė Nr. 2 *Metinės grąžos, leistini metiniai nepastovumai ir metinės grąžos rizikos.*

Algoritmas	SP500	DJIA	$SP500^{-1}$	$DJIA^{-1}$
$ANTI^{1}$	39% ± 38% 93%	29% ± 36% 71%	44% ± 32% 124%	93% ± 31% 291%
Stopping	1498% ± 26% 5777%	1031% ± 24% 4299%	1118% ± 21% 5248%	935% ± 21% 4482%
Coef_opt	37% ± 37% 88%	28% ± 36% 66%		94% ± 31% 292%
Window_opt		10% ± 36% 17%		

Algoritmu „$ANTI^{1}$“ pažymime BAH_{30}(ANTICOR), kuris remiasi pastoviu investavimu, visų langų atžvilgiu, t.y. pastoviai pirkti ir parduoti investicijas algoritmu ANTICOR*w*, kur $2 \le w \le 30$. BAH – yra paprasčiausia akcijų pirkimo ir laikymo strategija („buy-and-hold“), naudojanti tam tikrą akcijų portfelį. „Stopping“ žymime srategiją, kai naudojamas ANTICOR algoritmo stabdymas, o „Coef_opt“ – ANTICOR algoritmas, kuris naudoja genetinį algoritmą (1) lygties koeficientams rasti. Ir galiausiai algoritmas pažymėtas „Window_opt“ yra ANTICOR algoritmas, naudojantis genetinį algoritmą optimalaus lango radimui.

Financiniuose skaičiavimuose standartinis rizikos matas yra lygus grąžos standartiniam nuokrypiui. Lentelėje Nr. 2 radome metines grąžas (annualized returns), leidžiamus metinius nepastovumus (respective annualized volatilities) ir metines grąžos rizikas (Sharpe Ratio). Metinė grąža yra randama pagal formulę [3]:

$$AnnRtn(r_1,\ldots,r_n)=\left(\prod_{i=1}^{n}(1+r_i)\right)^{1/NumYears}-1.$$

Rizika, t.y. leidžiamas metinis nepastovumas, yra skaičiuojamas pagal formulę:

$$AnnRisk(r_1,\ldots,r_n)=\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(r_i-\bar{r}\right)^2}\cdot\sqrt{252}.$$

Metinė grąžos rizika yra lygi

$$ShRatio=\frac{AnnRtn(r_1,\ldots,r_n)-4}{AnnRisk(r_1,\ldots,r_n)}.$$

7 IŠVADOS

Šiame darbe buvo bandoma modeliuoti akcijų portfelį, kurį randame ANTICOR algoritmo pagalba. Pasitelkus genetinį algoritmą buvo rasti dydžio $claim_{i \to j}$ koeficientai ir optimizuotas langas w. Geresni rezultatai gauti kai kuriais laiko momentais, nei kad buvo iki šiol. Taipogi, buvo bandoma sustabdyti ANTICOR algoritmo naudojimą ir gauti rezultatai buvo stebėtinai geri, nes neatsižvelgta į pirkimo – pardavimo kainą, finansinių resursų užšaldymo kainas. Reikalingi tolimesni tyrimai, kuriuose būtų atsižvelgta į tas situacijas.

8 LITERATŪRA

[1] Borodin A., El-Yaniv R., Gogan V. Can We Learn to Beat the Best Stock. *Journal of Artificial Intelligence Research 21*, 579-594, (2004)

[2] Holland J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[3] http://www.styleadvisor.com/support/statistics/annualized_return.html

9 PRIEDAI

Priedas Nr. 1 *Algoritmo, naudojančio genetinį algoritmą koeficientų radimui, kodas.*

```
clear all
data = load('C:\Matlab7\work\matlab-data\DJA_2.mat');
data = data.DJA_2;
[periods,stocks] = size(data);
window = 12;
earned = 1;
earned1 = 1;
portfolio = 1/stocks * ones(1,stocks);
portfolio1 = 1/stocks * ones(1,stocks);

init_coef = [1,1,1];
fprintf('Strategy: ANTICOR(%d)\n',window);
opts = gaoptimset('Generations',10);

for t=2*window:periods-1

        portfolio = anticor(data,window,t,portfolio,init_coef);
        portfolio1 = anticor(data,window,t,portfolio1,[1,1,1]);

FitnessFunction = @(g_coef)anticor_fitness(g_coef,data,window,t,portfolio);
        numberOfVariables = 3;
        [coef_opt,fval] = ga(FitnessFunction,numberOfVariables,opts);
        init_coef = coef_opt;

        ret = data(t+1,:)*portfolio';
        earned = earned*ret;

        ret1 = data(t+1,:)*portfolio1';
        earned1 = earned1*ret1;

        s(t-2*window + 1) = earned;
        s1(t-2*window + 1) = earned1;

        plot(s,'b-');
        hold on
        plot(s1,'r-');
        pause(0.1);
      fprintf('Day %d, Return: %f, Earned: %f \n',t-2*window+1,ret,earned);
        legend('a,b,c','1,1,1');

end
```

Priedas Nr. 2 *Algoritmo, naudojančio stabdymą, kodas.*

```
clear all
data = load('C:\Matlab7\work\matlab-data\SP_5.mat');
data = data.SP_5;
[periods,stocks] = size(data);

for window = 2:30
    earned = 1;
    portfolio = 1/stocks * ones(1,stocks);

    stop = 0;

fprintf('Strategy: ANTICOR(%d)\n',window);
for t=2*window:periods-1

     if (stop == 0)
         portfolio = anticor(data,window,t,portfolio);
     end

     ret = data(t+1,:)*portfolio';

      if (ret < 1)
          ret = 1;
          stop = 1;
      else
          stop = 0;
      end

       earned = earned*ret;

       p(window) = earned;

       plot(p);
    end

end
```

Priedas Nr. 3 *Algoritmo, naudojančio genetinį algoritmą w optimizavimui, kodas.*

```
clear all
data = load('C:\MatLab7\work\matlab-data\DJA_2.mat');
data = data.DJA_2;
[periods,stocks] = size(data);
windows = [12,13];
earned = 1;
earned1 = 1;
portfolio = 1/stocks * ones(1,stocks);
portfolio1 = 1/stocks * ones(1,stocks);

init_coef = [0.5];
opts = gaoptimset('Generations',50);

for t=2*max(windows):periods-1

        portfolio = init_coef(1)*anticor(data,windows(1),t,portfolio) +
    (1 - init_coef(1))*anticor(data,windows(2),t,portfolio);
        portfolio1 =  anticor(data,windows(1),t,portfolio1);

        init_coef = [0.5];
        FitnessFunction =
    @(init_coef)anticor_fitness(init_coef,data,windows,t,portfolio);
        numberOfVariables = 1;
        [coef_opt,fval] = ga(FitnessFunction,numberOfVariables,opts);
        init_coef = coef_opt;

        ret = data(t+1,:)*portfolio';
        earned = earned*ret;

        ret1 = data(t+1,:)*portfolio1';
        earned1 = earned1*ret1;



        s(t-2*max(windows) + 1) = earned;
        s1(t-2*max(windows) + 1) = earned1;

        plot(s,'b-');
        hold on
        plot(s1,'r-');
        pause(0.1);
fprintf('Day %d, Return: %f, Earned: %f \n',t-2*max(windows)+1,ret,earned);
        legend('Kombinuotas','Paprastas');

end
```

SUMMARY

The portfolio selection problem is a challenging problem for machine learning, online algorithms and of course, computational finance. In this work was used a portfolio selection algorithm, which does not try to predict winners. There were used a genetic algorithm and algorithm stopping, trying to optimize the ANTICOR algorithm.