VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Dimitrianas Kondrašovas

VEKTORINIO OPTIMIZAVIMO UŽDAVINIŲ TYRIMAS MOBILIOJE APLINKOJE

INVESTIGATION OF VECTOR OPTIMIZATION PROBLEMS IN MOBILE ENVIRONMENT

Baigiamasis bakalauro darbas

Studijų programa: Inžinierinė informatika

Pakraipa: Inžinierinė

Vadovas habil. dr. prof. Jonas Mockus

Vilnius, 2005

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Inžinerinės informatikos katedra

ISBN ISSN
Egz. sk. 2
Data

Bakalauro studijų baigiamasis darbas
Vektorinio optimizavimo uždavinių tyrimas mobilioje aplinkoje

Dimitrianas Kondrašovas

Kalba
☒ lietuviu
☐ užsienio

Optimizavimo uždaviniai aktualūs įvairiose žmonių veiklos srityse. Kasdieninėje veikloje dažnai optimizuojama intuityviai, nesinaudojant jokia teorija. Tačiau svarbesniais atvejais stengiamasi uždavinį suformuluoti tiksliai ir jį spręsti tam pasirinktais metodais naudojant programine įrangą.

Plečiantis belaidžių technologijų rinkai, naudinga susipažinti su neseniai pasirodžiusia Java 2 Micro Edition (J2ME) platforma, kuri leidžia programuotojams kurti įspūdingas programas įvairiems mobiliesiems prietaisams, tarp jų ir kišeniniams kompiuteriams.

Šio darbo tikslas: susipažinti su vektorinių optimizavimų, bei sukurti programą, Java 2 Micro Edition mobilioje aplinkoje, kuri išspręstų vektorines optimizacijos uždavinį keliais metodais.

Vilnius Gediminas technical university
Fundamental science faculty
Engineering informatics department

ISBN ISSN
2 copies
Data

Bachelor studies diploma work
Investigation of vector optimization problems in mobile environment

Dimitrianas Kondrashovas

Language
☐ lithuanian
☒ foreign

Optimization tasks are urgent in different spheres of public work. It‘s often optimized intuitively, not using anytheories in every day activity. Nevertheless, in more important case one is doing the best to formulate the task exactly and to solve it using the chosen methods and programme installation.

The market of wireless technologies is spreading, so it is useful to become acquainted with Java 2 Micro Edition(J2ME) platform which appeared recently and allowed programmers to create impressive programmes for different mobile apparatus and for pocket computers as well.

The aim of this work: to get acquinted with vectorial optimization and to create programme(midlet) in Java 2 Micro Edition environment in order to solve vectorial optimization task in a few methods.

Turinys

Įvadas 5
1. Optimizacijos įvadas 6
1.1 Optimizavimo uždaviniai 6
1.2 Uždavinio formulavimas 6
1.3 Optimizavimo apibrėžimas 7
1.3 Optimizavimo pavyzdžiai 9
1.4 Metodų efektyvumas 10
2. Daugelio kriterijų optimizavimas 11
2.1 Išrinkimas pagal intuicija 11
2.2 Pareto optimalumas 11
2.3 Vektorinė optimizacija su svoriais 15
2.4 Nuosekli kriterijų analizė 18
2.5 Metodų aptarimas 20
2.7 Vektorinės optimizacijos uždavinys 21
3. Java 2 Micro Edition 23
3.1 Ateities platforma 24
3.2 MIDP ir CLDC 25
3.3 Java Virtual Machine 26
3.4 Saugi platforma 26
3.5 J2ME palyginimas su BREW platforma. 27
3.6 J2ME priklausomybė nuo techninių įrenginio parametrų 28
3.7 Pagrindiniai J2ME privalumai 29
3.8 Mobilaus telefono emuliatoriai 29
3.9 Kodėl būtent J2ME? 31
4. Best Mobile Plans J2ME programa 32
4.1 Panaudota programinė įranga ir reikalavimai 32
4.3 Programos Best Mobile Plans kūrimo aplinka 34
4.4 Programoje panaudoti J2ME paketai 36
4.4 Programos aprašas 36
4.5 Programos testavimas mobiliose aplinkose 41
5. Išvados ir Pasiūlymai 47
6. Literatūros sąrašas 48
7. Priedai 49
1 PRIEDAS 49
2 PRIEDAS 51

Įvadas

Kiekvienas žmogus, nepriklausomai nuo jo specialybės, realiam gyvenime susiduria su optimizavimu. Projektuotojai, siekia maksimalaus efektyvumo. Verslininkai ir vadybininkai stengiasi darbą organizuoti taip, kad būtų gautas maksimalus pelnas. Šeimininke, atėjusi į turgų, stengiasi pirkti prekę ten kur pigiau. Beveik kasdien kiekvienas žmogus susiduria su optimizavimo problemomis. Dažniausiai optimizuojama pasitelkiant savo intuicija. Tačiau, kai atvejis būna svarbesnis ir sudėtingesnis, stengiamasi uždavinį suformuluoti tiksliai, atsižvelgiant į apribojimus, ir išspręsti jį tam pasirinktais metodais.

Šiame darbe nagrinėjami vektorinio optimizavimo uždaviniai, bei galimi optimizavimo užduoties sprendimai, keliais vektorinės optimizacijos metodais, mobilioje aplinkoje.

Šiuo metu turbūt kiekvienas žmogus neįsivaizduoja savo gyvenimo be mobiliojo telefono. Viskas apsivertė aukštyn kojomis rinkoje pasirodžius su belaidė "Java" (J2ME) suderinamiems telefonų modeliams. Be to, kaip iš gausybės rago pasipylė nepriklausomų programinės įrangos gamintojų sukurtos programos, sėkmingai sulaužiusios visus įmanomus apribojimus.Įvairūs žaidimai, žodynai, žemėlapiai, pasaulio laiką rodantys laikrodžiai, kelionių gidai, MPEG grotuvai, vaizdo ir melodijų "redaktoriai", pagalbiniai sistemos valdymo įrankiai , visavertės interneto naršyklės, valiutos ir matavimo vienetų konverteriai, programos darbui su biržų informacija, mobiliosios elektroninės lentelės ir duomenų bazės, galimybė peržiūrėti trimatę grafiką bei diagramas – visa tai šiandien gali turėti kiekvienas mobiliojo ryšio vartotojas, turintis tobulesnį telefoną su integruota "Java" technologija. Todėl atsižvelgiant į technologijos Java 2 Wireless naujumą bei patogumą(mobilųjį telefoną visada galima nešioti su savimi) labai naudinga būtų su ją susipažinti, taip pat sukurti programą skirta mobiliajam telefonui, kuri leistų išspręsti tam tikra aktualia problemą realiu laiku, keliais vektorinės optimizacijos metodais, bei palyginti kaip sukurta programa veiktų įvairiuose „Java“ palaikančiose telefono modeliuose. Tačiau prieš tai reikėtų trumpai susipažinti su optimizavimo metodų apibrėžimais, metodais ir pilnai suprasti kas tai yra mobili aplinka.

1. Optimizacijos įvadas

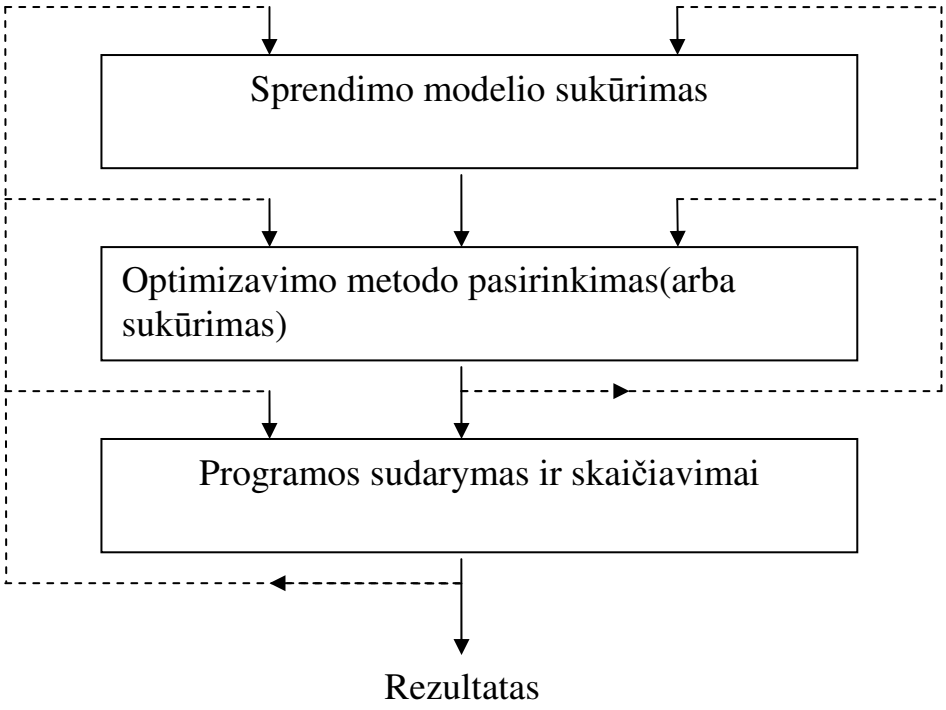
1.1 Optimizavimo uždaviniai

Dažnai realus optimizavimo uždavinius yra sunku suformuluoti ir išspręsti. Sunkumai iškyla ne tik sudarant uždavinio sprendimui reikalinga modelį, bet ir ieškant optimalias uždavinio parametrų reikšmes. Tai kartais reikalauja labai didelių skaičiavimų. Informatikos, vadybos, ekonomikos bei kitų sričių specialistai, sprendžiantis šiuolaikines problemas, gan dažnai susiduria su sudėtingais optimizavimo uždaviniais, kurių sprendimas reikalauja kiek gilesnio, negu vien metodo idėja, optimizavimo teorijos suvokimo. Todėl kiekvienam uždavinio sprendimui reikalinga gerai suprasti, uždavinio sprendimui parinkto metodo teorija ir jo efektyvumą.

Optimizavimo teorija ir optimizavimo metodų bei juos realizuojančios programinės įrangos kūrimas yra aktyviai vystoma mokslo ir technikos šaka. Programinės įrangos rinkoje egzistuoja daugybė komerciškai platinamų bei nemokamų programinių produktų, kurie skirti programinės įrangos kūrimui. Todėl pasirenkant kūrimo priemonę, optimizavimo uždavinių realizavimui, labai naudinga būtų atsižvelgti į jos naudojimo patogumą ir naujumą.

1.2 Uždavinio formulavimas

Kiekvieno uždavinio formulavimas yra labai svarbus etapas, nuo kurio priklauso ir uždavinio išsprendžiamumas, bei gautų rezultatų patikimumas ir racionalumas. Sprendžiant optimizavimo uždavinį kompiuteriu labai patogu būtų susidaryti uždavinio modelį, tam kad būtų patogiau pereiti prie tolimesnių žingsnių. 1 paveiksle matome kaip galima būtų schemos pagalba pavaizduoti uždavinio kūrimo etapus[2].



1 pav.Optimizacijos uždavinio kūrimo etapai.

Sudarant uždavinio sprendimo modelį reikalingos įvadinės teorijos žinios. Kadangi sudarant modelį beveik visada tenka rinktis kompromisą tarp modelio sudėtingumo ir adekvatumo, svarbu žinoti kuo daugiau apie metodus, kuriais bus sprendžiamas uždavinys, galimybes ir bendras savybes.

Sparčiai augant technologijoms bei esant šiuolaikiniam mokslo lygiui efektyvūs sprendimo metodai žinomi toli gražu ne kiekvienam optimizacijos uždaviniui. Todėl formuluojant uždavinį gerai būtų atsižvelgti į numatomų naudoti optimizavimo metodų efektyvumą.

Optimizavimo uždaviniais domėjosi, juos tyrinėjo ir metodus jiems kūrė tokie įžymūs mokslininkai kaip Newton, Lagrange, Euler ir daugelis kitų. Optimalumo principais išreiškiami svarbūs ekonomikos, vadybos, bei daugelio kitų sričių dėsniai, todėl šie principai labai naudingi ir svarbus mokslui.

1.3 Optimizavimo apibrėžimas

Optimizavimu vadinama tokio, uždavinio apibrėžtoje aibėje, elemento paieška, kuriam kriterijaus reikšmė būtų minimali (arba maksimali).

Šiuo baigiamojo darbo užduotyje, kurios praktinė dalis bus apžvelgta žemiau, naudosime minimizavimą, t.y. daugelio elementų paiešką, kurių kriterijų reikšmės bus minimalios.

Dažnai nagrinėjamoji aibė yra *n* - matės Euklido erdvės poaibis[1]. Tada galima kalbėti apie parametrų (vektoriaus) su minimalia kriterijaus reikšme paiešką. Parametrų vektoriaus $X \in R^n$ komponentės x_i, $i = 1, ..., n$, vadinamos uždavinio kintamaisiais, o kriterijų apibrėžianti funkcija $f(X)$ vadinama tikslo funkcija. Kintamieji turi patenkinti uždavinio formulavimą atitinkančias sąlygas.

Pavyzdžiui, jei kintamasis x_i reiškia kažkokio objekto kainą, jis negali būti neigiamas. Todėl formuluojant uždavinį reikia atsižvelgti į naudojamų parametrų apribojimus. Pavyzdžiui, jei nurodytą, kad kintamasis x_i kinta $0 \le x_i \le 5$ tai akivaizdžiai matysime, kad apatinis rėžis bus lygus 0, o viršutinis 5.

Kintamieji gali būti susieti lygybėmis ir nelygybėmis, išreikiančiomis turimus apribojimus, dydžius, reikalavimus ir pan. Pavyzdžiui, jei erdvės *Ra* poaibis *A,* kuriame patenkinamos uždavinyje apibrėžtos sąlygos kintamiesiems, vadinamas leistinąja sritimi. Leistinoji sritis paprastai apibrėžiama lygybėmis ir nelygybėmis *ribojimų funkcijų* reikšmėms, pvz.:

$$A = \{X \in R^n : p_1(X) = 0, p_2(X) \ge 1\}.$$

Minimizavimo uždavinį galima būtų užrašyti taip:

$$\min_{X \in A} f(X)$$

o jo sprendimas reiškia minimalios tikslo funkcijos reikšmės $f_* \le f(X),\ X \in A$ ir minimumo taško $X_* \in A,\ f(X_*) = f_*$ suradimą. Kad matematiškai suformuluotas uždavinys turėtų prasmę, reikia garantuoti sprendinio egzistavimą. Tam užtenka minimalių prielaidų, kad sprendinys egzistuoja. Uždavinio sprendinio egzistavimui taip pat pakanka, kad leistinoji aibė būtų baigtinė.
Reikėtų pasidomėti ar pasirinktas minimumo taško radimo uždavinys yra korektiškas, t.y. uždavinio sprendinys gali stipriai pasikeisti duomenims pasikeitus visai nežymiai.

Praktiškai priimtina ir teoriškai korektiška būtų tokia uždavinio formuluotė: rasti tikslo funkcijos reikšmę f ir ją atitinkanti leistinosios aibės tašką, kuriame tikslo funkcijos reikšmė skiriasi nuo minimalios ne daugiau negu leistina tolerancija. Mes darysime prielaidą nagrinėjamieji minimizavimo uždaviniai yra korektiški. Tada minimumo ir minimumo taško suradimo uždaviniai yra ekvivalentūs.

Iškyla natūralus klausymas: Kaip pritaikyti ir pasinaudoti optimizavimo teorija iškilus būtinybei, sprendžiant optimizacijos uždavinius? Norint taisyklingai suformuluoti uždavinį, reikia sudaryti taikomojo uždavinio modelį[1]:

- Išsiaiškinam, kokį kriterijų reikėtų optimizuoti, taip pat nustatome ribojimus, t.y. kokie uždavinio parametrai gali būti keičiami ir kokioje srityje jie gali kisti. Reikia apibrėžti kintamųjų vektorių X, tikslo funkcija *f(X)* ir leistiną sritį A. Kai panaudosime pasirinkta optimizavimo metodą, gausime optimalių parametrų vektorius X_*, kurios ir reikės praktiškai pasirinkti įgyvendinant uždavinio sprendinį.

- Kiekvienoje srityje, tam kad pilnai išspręsti uždavinį reikalingos žinios. Taip ir čia, tam kad būtų galima sudaryti algoritmą, tikslo funkcijos $f(X)$ ir ribojimus apibrėžiančių funkcijų $g_i(X)$ reikšmėms apskaičiuoti, reikalingos žinios apie taikomąją sritį.

Remiantis optimizacijos modelio rezultatais ir aprašymais galėsime parinkti optimalų metodą optimizacijos uždavinio sprendimui[1].

1.3 Optimizavimo pavyzdžiai

Pradėkime nuo labai paprasto uždavinio[1]. Kokia turėtų būti stačiakampio gretasienio formos dėžė, kad vienetiniam paviršiaus plotui jos tūris būtų maksimalus? Pažymėję dėžės ilgį, plotį ir aukštį x_1, x_2, x_3, tikslo funkciją galėsime užrašyti formule

$$f(X) = x_1 x_2 x_3,$$

ir leistinąją sritį trimis nelygybėmis ir viena lygybe

$$A = (X \in R^3 : x_i \geq 0,\ 2x_1x_2 + 2x_1x_3 + 2x_3x_2 = 1\}.$$

Modelis čia labai paprastas: vos kelios analitinės formulės. Uždavinys tikrai turi sprendinį, nes tikslo funkcija yra tolydi, o leistinoji sritis aprėžta ir uždara. Uždavinio kintamųjų prasmė akivaizdi: gaminamos dėžės matavimai. Ar išsprendus uždavinį gautas rezultatas bus naudingas praktiniams taikymams?

Taip, jei medžiagos taupymas yra pagrindinis tikslas: su duotu medžiagos, pvz. medienos, kiekiu galėtume pagaminti maksimalaus tūrio dėžę. Bet šis uždavinio formulavimas ignoruoja daugelį aplinkybių, kurios gali būti svarbios praktikoje. Pavyzdžiui, dažnai reikia ne vienos, o daugelio dėžių, mediena reikia apdirbti iš tam tikros formos ir minimizuoti pjaustymo bei apdirbimo atliekas, dėžes reikia vežti tam tikro dydžio konteineriuose ir pan. Atsižvelgiant i konkretaus taikymo aplinkybes, tikriausiai reikėtų tikslinti uždavinio modelį: atsižvelgti į daugiau ribojimų, ar net keisti tikslo funkciją. Daugelyje praktinių optimizavimo uždavinių modelio sudarymas yra gan ilgas procesas, kuriame dalyvauja suinteresuotų organizacijų specialistai ir vadybininkai.

Daugelyje sričių tenka minimizuoti išteklių, ypač brangiųjų, sunaudojimą. Kartais gali atsitikti taip, kad iki sprendimo priėmimo, pasirinktų pirkti, produktų, parametrų arba išteklių kainos bus nebe tokios,

kokios buvo anksčiau. Gali atsirasti naujų technologijų. Gali tekti atsižvelgti į faktorius, kuriu anksčiau nebuvo, nes kokia nauda iš labai geros plano, kurio realiai nebus įmanoma įgyvendinti.

Optimizavimo uždavinio modelio sudarymas ir suformuluoto optimizavimo uždavinio sprendimas gali būti sudėtingas ir pareikalauti tam tikrų išlaidų. Tačiau rinkos sąlygomis optimizuoti ir pačius gaminius, ir gamybos procesą, ir firmų vadybą būtina, nes konkuruoti tenka su tais, kurie optimizuoja. Tai parodo didele optimizacijos svarbą šiuolaikinėje visuomenėje.

1.4 Metodų efektyvumas

Tam kad pasirinkti metodą, skirtą uždavinio sprendimui įgyvendinti, reikalinga atsižvelgti į pasirinkto metodo efektyvumą.

Ne visada paprasta net patikrinti, ar duotasis taškas $X \in A$ yra nagrinėjamojo uždavinio minimumo tašku, ar ne, ir tik labai specialiais atvejais optimizavimo uždavinį pasiseka išspręsti analitiškai, t.y. formule užrašyti sprendinį. Minimumo apibrėžimas bendru atveju nelabai kuo gali padėti, nes kontinualios(t.y. aibės visu reikšmių negalima patikrinti netgi atlikus be galo daug veiksmu) leistinosios srities atveju neįmanoma palyginti nagrinėjamos tikslo funkcijos reikšmės su visomis kitomis. Kartais potencialių sprendinių aibė yra baigtinė, nors leistinoji sritis ir kontinuali. Taip yra, pavyzdžiui, tiesinio programavimo uždaviniuose, kuriuose leistinoji aibė yra kontinuali. Surasti optimalų sprendinį baigtinėje potencialių sprendinių aibėje galima atlikus pilną aibės peržiūrą, t.y. baigtiniu žingsnių skaičiumi, nors pastarasis gali būti ir labai didelis[1].

Kai optimizavimo uždavinys kokiu nors metodu išsprendžiamas baigtiniu žingsnių skaičiumi N, metodo efektyvumas gali būti vertinamas algoritmų sudėtingumo teorijos kriterijais. Įvertinama žingsnių skaičiaus N priklausomybė nuo n ir kitų uždavinio parametrų, ir šios priklausomybės tipas apsprendžia metodo sudėtingumo klasę. Kuo greičiau auga N, tuo sudėtingesnis metodas, arba tuo menkesnis jo efektyvumas.

2. Daugelio kriterijų optimizavimas

2.1 Išrinkimas pagal intuicija

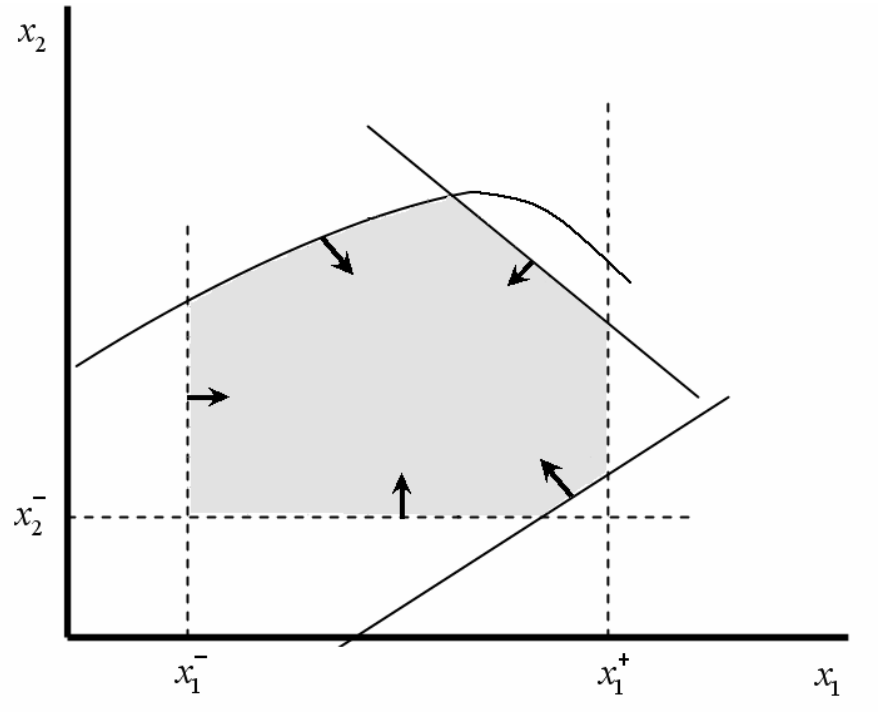
Svarbus optimizacijos uždavinio apibendrinimas, kuriam ir skirsime pagrindinį dėmesį, susijęs su būtinumu optimizuojant atsižvelgti į daugelį kriterijų. Tokius uždavinius nagrinėja daugelio kriterijų optimizacijos teorija.

Daugelio kriterijų uždaviniai iškyla įvairiose sprendimų priėmimo situacijose. Panagrinėkime projektavimą. Kiekvienas projektavimo etapas baigiamas vieno arba kelių variantų išrinkimu tolesniam darbui. Pavyzdžiui, žmogui pateikiami penki galimi mobiliųjų planų variantai X_i, i = 1, ..., 5, besiskiriantys kriterijų vektoriaus $F(X) = (f_1(X), ..., f_5(X))$ reikšmėmis. Palyginęs planus, žmogus savo išvadas dažnai formuoja, pavyzdžiui, šitaip: variantas X_1 nepriimtinas dėl per didelės kriterijaus f_1 (kainos) reikšmės, o X_2 - dėl per didelės f_2 reikšmės. Variantas X_3 atmetamas, nes beveik pagal visus kriterijus blogesnis už X_2, tik pasižymi mažu f_5 minimaliu plano mokesčiu, o X_5 geresnis pagal visus kriterijus už X_3 bet blogesnis už X_2.

Išnagrinėjęs visus variantus, žmogus bus priimtas racionalus sprendimas. Tačiau ar tie keli variantai atspindi visas galimybes? Iš kitos pusės, žmogus gali neaprėpti visų galimų variantų, jei jų yra keli šimtai, ypač tuomet, kai kiekvienas variantas apibūdinamas dešimtimis kriterijų. Pavyzdžiui, jei pvz. su mobiliaisiais planais planų ir kriterijų būtų daugiau už 10, išrinkti racionaliai būtų žymiai sudėtingiau ir, atmetus aiškiai nepriimtinus, išrinkti tam tikrą potencialiai geriausių variantų skaičių. Todėl efektyviai galima dirbti tik panaudojus optimizavimo metodus. Racionalaus išrinkimo uždavinį teoriškai galima pakeisti naudingumo funkcijos optimizavimo uždaviniu. Jei sudaryti naudingumo funkciją dėl įvairių priežasčių sunku, tai galime pasinaudoti alternatyviais metodais[3].

2.2 Pareto optimalumas

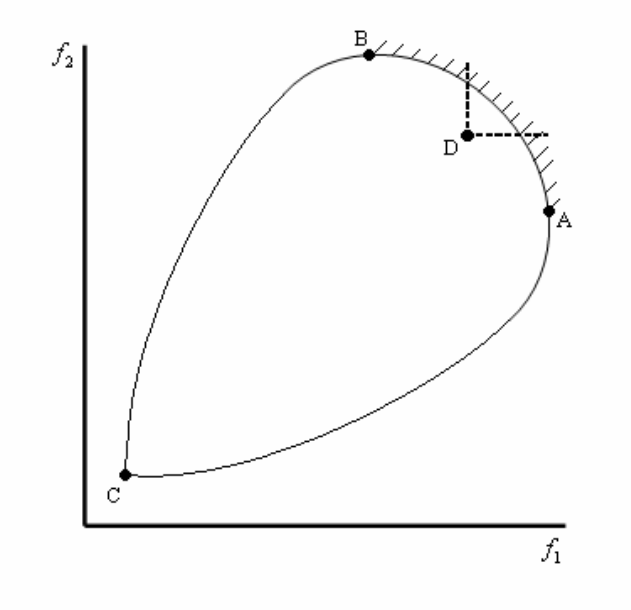
Tarkime turime vektorių $X = (x_1, ..., x_n)$ kuris turi n keičiamų parametrų. Jis parenkamas leistinojoje srityje(arba aibėje) $A \in R^n$. Kai kuriems kintamiesiems mes galime apibrėžti apatinę, kai kuriems viršutinę, o kai kuriems ir abi ribas. Pavyzdžiui, leistinoji sritis 2 paveiksle apibrėžta trimis paprastaisiais, vienu netiesiniu ir dviem tiesiniais ribojimais[1][3].



2 pav. Leistinosios aibės pavyzdys

Kiekvienam vektoriui X apibrėžiamas kriterijų vektorius $(f_1, ..., f_m) = F(X)$. Vektoriui X kintant aibėje *A* vektorius *F* perbėga kriterijų reikšmių aibę Φ. Jei kriterijus būtų tik vienas (tolydinis X atžvilgiu), tai Φ sutaptų su intervalu, kurio pradžią ir galą nustato kriterijaus funkcijos (vieno kriterijaus uždaviniuose vadinamos tikslo funkcija) minimumas ir maksimumas. Jei yra daug kriterijų, tai jų minimumų taškai paprastai nesutampa.

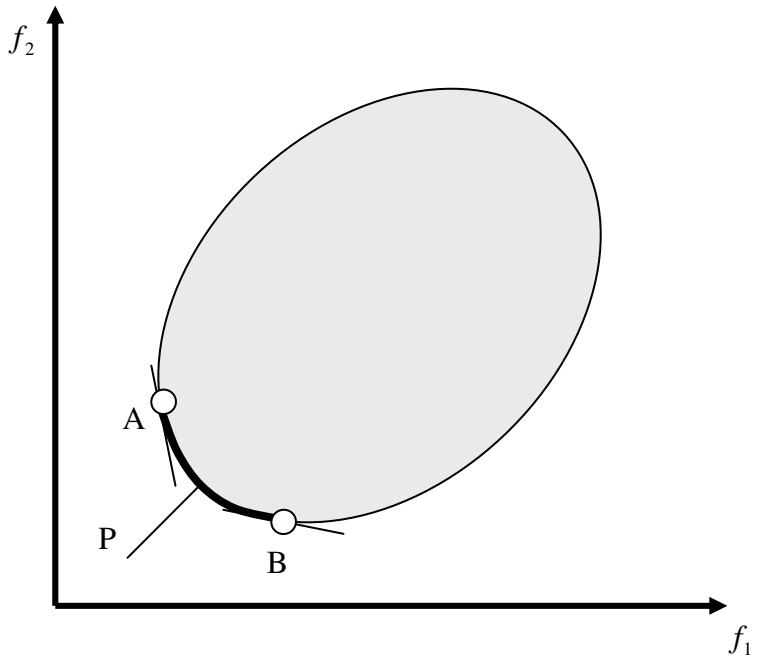
Pavaizduokime dviejų kriterijų reikšmių aibėje(3 pav.), pirmojo kriterijaus maksimumo tašką *A* ir antrojo kriterijaus maksimumo tašką *B*, kurie nesutampa. Tačiau sutampa abiejų kriterijų minimumai taške *C*[1].



3 pav. Kriterijų reikšmių aibės pavyzdys

Daugelio kriterijų uždavinių teorijoje paprastai nagrinėjamas visų kriterijų maksimizavimas arba minimizavimas. Vieno kriterijaus optimizavimo uždaviniuose tikslo funkciją minimizavome. 3 paveiksle iliustruojamas dviejų kriterijų maksimizavimo uždavinys gali atrodyti prieštaringas. Jei kaip sprendinį imsime tašką *A,* tai gerokai nukrypsime nuo maksimalios antrojo kriterijaus reikšmės, ir atvirkščiai. Todėl reikia ieškoti kompromisinio varianto. Kita vertus, *D* taip pat negali būti sprendiniu, nes visi taškai pažymėtame sektoriuje yra geresni už tašką *D* abiejų kriterijų atžvilgiu. Taškai, iš kurių negalima pajudėti pagerinant abu kriterijus, sudaro aibės Φ šiaurės - rytų ribą, kuri pavyzdžiui užbrūkšniuota 3 paveiksle. Toks poaibis Φ vadinamas *Pareto arba kompromisų aibe*. Jei abu kriterijai lygiaverčiai, tai nė vieno iš šio poaibio taškų negalime išskirti, nes kiekvieną iš jų vienodai tinkamas kaip kompromisinis variantas. 3 paveiksle kiekvienas užbrūkšniuotas kreivės taškas nepagerinamas.

Išanalizuokime dar vieną situaciją pavaizduota 4 pav.[2]:



4 pav. Pareto taškai kai egzistuoja 2 kriterijai.

Pilka sritis parodo tas dviejų optimizavimo kriterijų f_1 ir f_2 reikšmes, kurios patenka į vektoriaus X kintamųjų leistinąją sritį. Paėmę tašką pilkos srities viduryje, pamatysime, kad kiekvienas taškas, kuris yra kairiau už vidurio tašką, kriterijui f_2 nesikeičiant, yra geresnis pagal kriterijų f_1. Reiškia, kad šį tašką mes galėjome pagerinti ir jį negalime vadinti optimaliu. Jeigu paimsime tašką, kuris yra P kontūre, matysime, kad kaip mes jį nejudinsime kažkuris iš kriterijų didės. Iš šio pavyzdžio galima pamatyti, kad egzistuoja gali egzistuoti daugybė Pareto taškų, kurie yra pasiskirstę tarp taško A ir B kontūre P[1][3].

Projektuotoją domina, kaip surasti variantą, atitinkantį nepagerinamą kriterijų vektorių. Taškas X_* vadinamas *efektyviuoju sprendiniu* (optimaliu Pareto prasme), jei nelygybė $F(X) \le F(X^*)$ ekvivalenti lygybei $X = X^*$, t.y. neegzistuoja toks $X \in A$, kuriam

$$f_i(X) \le f_i(X^*),\ i = 1, ..., m \qquad (2.1)$$

ir bent vienam *j,*

$$f_j(X) < f_j(X^*). \qquad (2.2)$$

Efektyvaus sprendinio kriterijų vektorius nepagerinamas, ir atvirkščiai: kiekvieną nepagerinamą kriterijų (Pareto) vektorių atitinka efektyvus sprendinys[6].

Pavyzdžiui, jei pateikiami tris galimi mobiliųjų planų variantai X_i, i = 1, ..., 3, besiskiriantys kriterijų vektoriaus $F(X) = (f_1(X), ..., f_5(X))$ reikšmėmis, kuriu reikšmes:

$F(X_1) = (0.2,0.1,0.3,0.2,0.4)$

$F(X_2) = (0.1,0.2,0.3,0.4,0.5)$

$F(X_3) = (0.4,0.1,0.3,0.2,0.4)$

Matysime, kad minimizavimo atveju, skaitydami, kad kuo mažesne kriterijaus reikšmė tuo geriau, gausime kad X_1 ir X_2 Pareto, o X_3 ne. Tai nesunku įrodyti, nes pagal Pareto apibrėžimą, geresnis tas kuris pagal visus parametrus nėra blogesnis ir nors pagal vieną parametrą yra geresnis.

2.3 Vektorinė optimizacija su svoriais

Racionalus išrinkimas gali būti pakeistas naudingumo funkcijos maksimizavimu(arba minimizavimu)[1][6]. Tačiau ir nežinant šio rezultato atrodo, kad gan natūralu pakeisti daugelio kriterijų uždavinį vienakriteriniu, sudarytu susumavus su svoriais visus uždavinio kriterijus

$$f(X) = \sum_{i=1}^{m} a_i f_i(X),\ \sum_{i=1}^{m} a_i = 1, a_i \geq 0. \qquad (2.3)$$

Pavyzdžiui, jei paimsime tą pati pavyzdi su mobiliaisiais planais, kuris paminėtas aukščiau, tai matysime, kad išrenkant planus pagal Pareto optimalų kriterijų bus atmesti visi neracionalus planai ir po to iš optimalių Pareto planų galima bus išrinkti sau tinkamą. Tuomet iškyla klausymas: Kaip tą patį uždavinį galima būtų išspręsti pritaikant svorius?

Jei paimsime realia situaciją tai matysime, kad ne visada žmogui yra svarbus tam tikri parametrai. Pavyzdžiui, perkant mašiną vienam svarbu, kad būtų galingesnis variklis, kitam kad mašina atrodytų gražiai ir pan. Todėl grįžtant prie mobiliųjų planų uždavinio galima padaryti išvadas, kad jeigu vienam svarbiau mažesne kainą skambinant, tai kitas gali rinktis visai pagal kitus kriterijus, pvz. jam bus svarbiau žinutės kainą į kitus tinklus. Tuomet programoje, kuri išrenka optimalų variantą pirmas galėtų nurodyti didesnį svorį skambučio kainai, o antras žinutės. Jeigu sprendžiant pagal Pareto optimalumą (2.1) (2.2) bus atmesti tik neracionalus planai, tai pagal vektorinę optimizacija su svoriais galima pritaikyti mobiliųjį planą pagal konkrečias užgaidas.

Panagrinėkime maksimizavimą su svoriais. Kokiu pagrindu gautąją funkciją f(X) galima laikyti pradinio uždavinio naudingumo funkcija? Ar teoriškai pagrįstas daugiakriterio uždavinio keitimas funkcijos *f(X)* maksimizavimo uždaviniu?

Paprastai, psichologiniu eksperimentu pavyksta nustatyti tik nedideliam alternatyvų skaičiui. Todėl galima tikėtis parinkti svorius (2.3) taip, kad funkcijos *f(X)* reikšmių santykiai atitiks eksperimentinius duomenis. Kita vertus, kartais kriterijų svorius galima nustatyti atsižvelgiant į kriterijų svarbą. Panagrinėsime svertinės kriterijų sumos savybes[3]:

1 teorema. Jei aibė *A* yra iškili, funkcijos $f_i(X)$ - įgaubtos (iškilos į viršų) ir X^0 - efektyvaus sprendinio taškas, tai egzistuoja tokie $a_i^0 \geq 0, \sum_{i=1}^{m} a_i^0 = 1,$ kad

$$\max_{X \in A} \sum_{i=1}^{m} a_i^0 f_i(X) = \sum_{i=1}^{m} a_i^0 f_i(X^0). \qquad (2.4)$$

Šis S.Karlin'o rezultatas teoriškai pagrindžia svertinės svorių kompozicijos taikymą.

2 teorema. Jei kai kuriems $a_i \geq 0$, $i = 1, ..., m$,

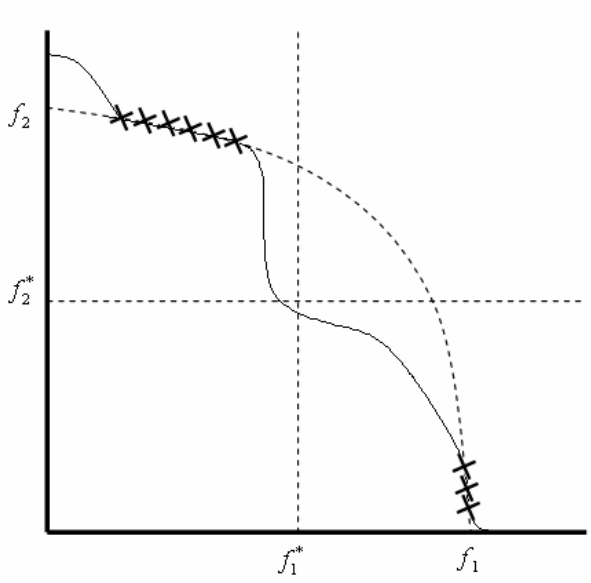
$$\sum_{i=1}^{m} a_i f_i(X^*) = \max_{X \in A} \sum_{i=1}^{m} a_i f_i(X) \qquad (2.5)$$

tai *X** yra efektyvus sprendinys.

Iš šių teoremų matome, kad maksimizuojant svertinę kriterijų sumą visada gaunamas efektyvusis sprendinys. Atitinkamai parinkdami svorius, galime gauti bet kurį efektyvųjį sprendinį. Gali susidaryti įspūdis, kad naudodamiesi šia kriterijų kompozicija galėsime išspręsti visas daugiakriterio optimizavimo problemas. Tačiau iškilumo reikalavimas Karlin'o teoremoje susiaurina jos pritaikymo sritį, nes kriterijų iškilumas projektavimo uždaviniuose gali būti abejotinas ir tik atskirais atvejais įrodomas. Jei iškilumo sąlyga ir priimtina, tai dar lieka neaišku, kokia yra priklausomybė tarp parenkamų svorių ir juos atitinkančių Pareto aibės taškų. Jei apie kriterijus turime papildomos informacijos, tai kaip ją įvertinti parenkant svorius? Pavyzdžiui, reikia gauti *N* tolygiai pasiskirsčiusių Pareto aibėje taškų. Kaip parinkti svorius? Parenkant svorius atsitiktinai su tolygiu pasiskirstymu, juos atitinką kriterijų vektoriai gali būti pasiskirstę Pareto aibėje labai netolygiai. 4 paveiksle pavaizduotas gana tipiškas pavyzdys: atskiruose Pareto aibės poaibiuose taškai išsidėstę tankiai, o kai kur jų visai nėra. Todėl lieka neaišku, kokia yra Pareto aibės forma. Naudodamiesi turimais duomenimis negalime atsakyti, ar egzistuoja tokie kriterijų vektoriai, kad $f_i > f_i^*$ [3]. Ekstrapoliuodami taip, kaip parodyta paveiksle brūkšnine linija, galėtume tikėtis, kad tinkamai parinkus svorius, gausime geresnį už (f_1^*, f_2^*) kriterijų vektorių. Tačiau gali

pasirodyti, kad tikroji riba yra tokia, kokią vaizduoja ištisinė linija, ir nėra vektoriaus su komponenčių reikšmėmis f_i^*.

Svertinė kriterijų kompozicija nėra panacėja, bet ji dažnai naudojama kuriant daugiakriterio optimizavimo metodus. Bendras kriterijus, gautas atlikus svertinės kriterijų sumos kompoziciją, pasižymi tuo, kad sumažėjus vienam kriterijui, tai kompensuojama, padidinus kitą. Kitu atveju, tai gali būti lyg ir atskirų pelno komponenčių suma: svarbu maksimizuoti arba minimizuoti bendrą pelną ir visai nesvarbu, kurios komponentės sąskaita jis gaunamas[3].



5 pav. Optimizuojant svertinę svorių kompoziciją, taškai Pareto aibėje išsidėsto netolygiai.

Esti uždavinių, kuriuose variantas nepriimtinas, jei bent vieno kriterijaus reikšmė lygi "nuliui". Čia nulis parašytas kabutėse todėl, kad natūraliame mastelyje nepriimtina kriterijaus reikšmė gali ir nebūti lygi nuliui. Tačiau, pakeitus skalės ataskaitos tašką, šią reikšmę galima laikyti nuline. Antra vertus, net ir labai geros kelių kriterijų reikšmės gali būti nuvertintos, kai vienas iš kriterijų artimas nuliui. Šiuo atveju variantas priimtinas tik tuomet, kai visų kriterijų reikšmės bent jau "neblogos" (anksčiau aprašytu svertinės sumos atveju variantas galėtų būti priimtinas ir tuomet, kai vieno kriterijaus reikšmė "labai bloga", t.y. nulinė). Tokias savybes turi kompozicija, išreiškiama kriterijų sandauga[1]:

$$f(X) = f_1(X) \cdot f_2(X) \cdot \ldots \cdot f_m(X). \qquad (2.6)$$

Jei funkciją f(X) logaritmuosime, tai gausime atskirų kriterijų logaritmų sumą. Todėl šios kompozicijos matematinės savybės yra analogiškos svertinės sumos savybėms.

Pateiksime dar vieną kriterijų kompozicijos pavyzdį[3]

$$f(X) = \min_i a_i \cdot f_i(X), \qquad (2.7)$$

kuria išreiškiamas "silpniausios grandies stiprinimo" principas. Juo dažnai naudojasi projektuotojai.

Kriterijų kompozicijos optimizavimas pagrindžiamas įvairiais euristiniais samprotavimais ir matematinėmis kompozicijos funkcijų savybėmis. Viena iš pagrindinių šio metodo problemų - kaip informaciją apie kriterijus išreikšti jų svoriais. Apibrėžus kompozicijos funkciją, uždavinys sprendžiamas įprastais optimizavimo metodais.

2.4 Nuosekli kriterijų analizė

Tyrimai parodė, kad žmonės alternatyvas išrenka nuosekliai analizuodami atskirus kriterijus. Pirmiausia atmetamos alternatyvos, netinkančios pagal svarbiausiąjį kriterijų, po to – pagal antrąjį ir t.t. Pavyzdžiui, pirkėjas gali rinkdamas prekę pirmiausia pažiūrėti kokia tos prekės kainą, o tik paskui atsižvelgti į kitus prekės parametrus ir jeigu kitas parametras jam nepatiks pirkėjas iš karto gali tą prekę atmesti į šalį.Tinkamiausi variantai išrenkami nuosekliai atmetant netinkamus atskirų kriterijų požiūriu. Panašūs rezultatai buvo gauti atliekant psichologinius eksperimentus. Pavyzdžiui, dalyviai turėjo išrinkti iš daugelio daiktų vieną, reikalingą eksperimento vedėjui. Spręsdamas šį uždavinį, eksperimento dalyvis užduodavo vedėjui klausimus apie reikalingo daikto požymius. Daugelis klausimų buvo formuojami taip, kad būtų galima atmesti visus daiktus, neturinčius reikalingo daikto požymio[3][1].

Surikiuojant kriterijus, maksimizavimo atveju, gauname leksikografinės optimizacijos uždavinį: variantas X_1 geresnis už X_2, jei $f_1(X_1) > f_1(X_2)$. Jei $f_1(X_1) = f_1(X_2)$, tai lyginama pagal antrąjį kriterijų ir t. t. Naudojant leksikografini išrinkimą, kai kuriuose čempionatuose išaiškinami čempionai, pavyzdžiui:

- čempionu tampa komanda, surinkusi maksimalų taškų skaičių (pergalė -1 taškas, lygiosios -1/2 taško, pralaimėjimas -0 taškų);

- jei kelios komandos surenka vienodą taškų skaičių, tai čempionu tampa ta komanda, kuri surinko daugiausia taškų tarpusavio susitikimuose;

- jei pagal pirmuosius du kriterijus kelios komandos yra lygiavertės, tai čempionu tampa ta, kurios įmuštų ir praleistų įvarčių santykis jų tarpusavio rungtynėse geriausias;

- jei kelios komandos lygiavertės pagal tris ankstesnius kriterijus, tai čempionas išaiškinamas atsižvelgiant į įvarčių santykį visose čempionato rungtynėse.

Kad kelios komandos būtų lygiavertės pagal kelis kriterijus nors ir mažai tikėtina, bet pasitaiko.

Leksikografinis optimizavimo uždavinys gali būti sprendžiamas nuosekliai maksimizuojant(minimizuojant) kriterijus siaurėjančiuose leistinos aibės poaibiuose $A_i \subset A$ [3]:

pirmasis poaibis:

$$A_1 = \{X : X \in A,\ f_1(X) = f_1^*\}, \max_{X \in A} f_1(X) = f_1^*,$$

antrasis poaibis:

$$A_2 = \{X : X \in A,\ f_2(X) = f_2^*\}, \max_{X \in A} f_2(X) = f_2^*,$$

m-asis poaibis:

$$A_m = \{X : X \in A,\ f_m(X) = f_m^*\}, \max_{X \in A} f_m(X) = f_m^*,$$

m-ojo maksimizavimo uždavinio maksimumo taškas ir bus sprendiniu. Jei kuriame nors žingsnyje aibę sudaro vienintelis taškas, tai jis ir yra leksikografinio uždavinio sprendinys. Leksikografiniai uždaviniai, matyt, natūraliausi tuomet, kai leistinoji (arba kriterijų reikšmių) aibė yra diskreti, nes jei A kontinua1i, o F(X) tolydinė, tai sudaryti poaibius A_i sudėtinga[1][3].

Aukščiau nurodytas būdas, skirtas leksikografiniam uždaviniui spręsti, susideda iš m etapų, kuriuose turi būti surastas funkcijos maksimumas aibėje, apibrėžtoje lygininiu ribojimu. Leksikografiniai uždaviniai su bega1inėmis kriterijų reikšmių aibėmis įdomūs tuo, kad ne visada egzistuoja naudingumo funkcija, atitinkanti leksikografinį preferencijos santykį. Todėl leksikografinis uždavinys ne visada gali būti pakeistas vienu optimizavimo (naudingumo funkcijos maksimizavimo) uždaviniu. Taigi, naudingumo teorijos prielaidos nėra universalios, o leksikografinis uždavinys iliustruoja atvejį, kai ga1i neegzistuoti uždavinį atitinkanti kriterijų kompozicija.

Leksikografinio uždavinio sprendimo ir daugelio kriterijų uždavinio pakeitimo įprastu matematinio programavimo uždaviniu metodai gali būti tam tikra prasme apjungti į vieną. Daugelyje praktinių uždavinių kriterijai gali būti surikiuoti (nors ir ne taip griežtai kaip leksikografiniu atveju). Sakykime, kad kriterijus tuo svarbesnis, kuo mažesnis jo numeris. Kartais lengviau suformuluoti ne ribojimus kriterijų reikšmėms, o nuolaidas d_i, kurios leidžiamas nuo maksimalių kriterijų reikšmių, siekiant pagerinti mažiau svarbių kriterijų reikšmes. Tuomet apskaičiuotas maksimalias svarbiausiųjų kriterijų reikšmes galima naudoti leistinajai sričiai apibrėžti žemesniojo rango kriterijų maksimizavimo uždaviniuose.

2.5 Metodų aptarimas

Daugiakriterių optimizavimo uždavinių klasifikacijos labai sąlyginės. Antra vertus, vartotojų informacija apie kriterijus labai įvairi, ir kiekvieno sudėtingo praktinio uždavinio sprendimas, galima sakyti, yra menas. Matyt, todėl net storose knygose, skirtose daugiakriterei optimizacijai, nėra vienareikšmių rekomendacijų, kaip pasirinkti metodą konkretiems uždaviniams spręsti[3].

Tam kad projektuotojas galėtų pagrįsti savo pasirinktą variantą, jis siekia sužinoti kriterijų kitimo ribas, tipines kriterijų reikšmių kombinacijas. Todėl, kai kriterijai yra neprieštaringi, pasiteisina parametrų erdvės zondavimo metodas.

Dažnai pasiteisina daugiakriterių uždavinių keitimas vieno kriterijaus optimizavimo uždavinių seka: tam tikras kriterijus pasirenkamas tikslo funkcija, visi kiti keičiami ribojimais. Taip buvo išspręsta nemažai įvairių sričių optimalaus projektavimo uždavinių. Nors šis metodas ir nėra labai gerai teoriškai pagrįstas, bet, remiantis sukaupta patirtimi, galima efektyviai panaudoti įprastus optimizavimo metodus.

Jei kriterijai gali būti interpretuojami kaip pelnas arba nuostoliai, tai tikėtina surasti bendrą jų matavimo skalę, įgalinančią naudotis svertinės sumos kompozicija.

Praktikoje iškylantys uždaviniai dažnai turi būti išspręsti iki griežtai nustatytų terminų. Todėl projektuotojui nebėra kada eksperimentuoti bandant įvairius metodus.

Pasirenkant metodą, rekomenduotume atkreipti dėmesį į šiuos aspektus:

- ✓ **Adekvatumas.** Daugiakriteriai uždaviniai paprastai sprendžiami dialoginiais metodais. Dialoge su kompiuteriu turi dalyvauti kompetentingas specialistas arba jų grupė. Svarbu užtikrinti, kad tai tikrai būtų tas dialogas, kuris numatytas metode. Reikėtų detaliai apgalvoti, ar tie klausimai, i kuriuos turės atsakyti ekspertai, yra būdingi jų veiklai, ar dialogas padės giliau suprasti uždavinį. Metodą galima vadinti tinkamu, jei ji naudodamas vartotojas dirba tikslingai, turima informacija naudojama racionaliai.

- ✓ **Naudojimo paprastumas**. Nepatyrusiam vartotojui yra sukurti specialūs dialoginių metodų variantai. Vėliau, įgijęs patirties, vartotojas gali dirbti naudodamas efektyvesnius metodo variantus, skirtus patyrusiam vairuotojui arba "meistrui" .To paties metodo variantų realizacijos skirtingoms vartotojų kategorijoms, gali gerokai skirtis.

- ✓ **Efektyvumas.** Metodų kūrėjų požiūriu tai pati svarbiausia metodo savybė, įrodoma teoriškai arba nustatoma eksperimentiškai. Ar metodas praktiškai efektyvus, galima sužinoti tik tada, kai jis yra pakankamai plačiai vartojamas, t. y. pakankamai paprastas ir priimtinas daugeliui vartotojų. Metodas laikomas praktiškai efektyviu, jei juo randami geresni variantai, negu vartojant kitus metodus.

2.7 *Vektorinės optimizacijos uždavinys*

Lietuvoje yra trys stambiausios ryšio bendrovės. „Tele2“, „Bitė GSM“ ir „Omnitel“ – konkuruojančios tarpusavyje. Šių bendrovių paslaugų spektras yra gana panašus. Kiekviena mobiliojo ryšio bendrovė didelį dėmesį skiria paslaugų kūrimui ir tobulinimui. Suprantama, tam kad suformuluoti optimizacijos užduotį, reikėtų peržvelgti su šių bendrovių paslaugomis susijusią informaciją.

Pirmiausia klientą domina siūlomi mokėjimo planai, pavyzdžiui pasirinkime šios:

- ✓ „Bitė GSM“ - *"Bite Privatiems2", "Bite Privatiems1", "Bite Eko", "Bite verslui", "Bite Verslui stac"*.
- ✓ „Tele2“ - *"Tele2 Joker Standartinis1", "Tele2 Joker Laisvalaikis", "Tele2 Joker Šeima", "Tele2 Verslas"*.
- ✓ „Omnitel“ - *"Omnitel Draugai", "Omnitel Mano", "Omnitel Lyderis20", "Omnitel Valdininkas20"*.

Kiekvienas mokėjimo planas turi bendrus parametrus su kainom, ką ir iliustruosime lentelėje:

Bendrovė	Plano pavadinimas	Parametrai										
		Dienos metu				Nakties metu				Kiti		
		Sk. savam tinkle (Lt)	Sk. kitame tinkle (Lt)	Sms savam tinkle (Lt)	Sms į kitus tinklus (Lt)	Sk. savam tinkle (Lt)	Sk. kitame tinkle (Lt)	Sms savam tinkle (Lt)	Sms į kitus tinklus (Lt)	Sms į užsienį (Lt)	Sk. savam rate (Lt)	Minimalus plano mokestis (Lt)
Bite GSM												
	Bite Privatiems2	0.39	0.69	0.15	0.15	0.39	0.69	0.15	0.15	0.45	0.39	19
	Bite Privatiems1	0.50	0.80	0.35	0.35	0.30	0.55	0.20	0.20	0.45	0.15	30
	Bite Eko	0.24	0.49	0.15	0.15	0.24	0.49	0.15	0.15	0.45	0.24	15
	Bite verslui	0.45	0.65	0.20	0.30	0.25	0.45	0.20	0.30	0.45	0.09	20
	Bite Verslui stac	0.25	0.46	0.20	0.30	0.18	0.32	0.20	0.30	0.45	0.09	150
Tele2												
	Tele2 Joker Standartinis1	0.25	0.65	0.12	0.12	0.25	0.65	0.12	0.12	0.15	0.25	15
	Tele2 Joker Laisvalaikis	0.59	0.59	0.12	0.12	0.25	0.25	0.12	0.12	0.15	0.59	30
	Tele2 Joker Šeima	0.25	0.65	0.12	0.12	0.25	0.65	0.12	0.12	0.15	0.13	15
	Tele2 Verslas	0.30	0.45	0.15	0.15	0.15	0.30	0.15	0.15	0.15	0.03	40
Omnitel												
	Omnitel Draugai	0.29	0.69	0.15	0.15	0.29	0.69	0.15	0.15	0.45	0.19	16
	Omnitel Mano	0.39	0.39	0.15	0.15	0.39	0.39	0.15	0.15	0.45	0.00	18
	Omnitel Lyderis20	0.50	0.70	0.25	0.35	0.30	0.50	0.25	0.35	0.45	0.10	20
	Omnitel Valdininkas20	0.50	0.60	0.25	0.35	0.30	0.40	0.25	0.35	0.45	0.10	20

1 Lentelė. Mobiliųjų planų parametrai.

Apžvelkime pasirinktus bendrus mobilių mokėjimo planų parametrus:

1. Skambučiai savam tinkle – tai yra minutės kaina skambinant savam tinkle. Sąvoka „savas tinklas“ reiškia, kad nurodyta skambučių kainą galios tik tada, kai mokėjimo plano turėtojas skambins žmogui, kuris turi mokėjimo planą priklausanti tai pačiai ryšio bendrovei.

2. Skambučiai kitame tinkle – tai yra minutės kaina skambinant į kitus tinklus. Sąvoka „kitas tinklas“ reiškia, kad nurodyta skambučių kainą galios tik tada, kai mokėjimo plano turėtojas skambins žmogui, kuris turi mokėjimo planą priklausanti kitai ryšio bendrovei.

3. Sms savam tinkle – tai yra trumpojo pranešimo(sms) kaina siunčiant jį savam tinkle.

4. Sms savam tinkle – tai yra trumpojo pranešimo(sms) kaina siunčiant jį į kitus tinklus.

5. Sms į užsienį – tai yra trumpojo pranešimo(sms) kaina siunčiant jį į užsienio tinklus.

6. Skambučiai savam rate – tai yra minutės kaina skambinant „savam rate“. Tai pavyzdžiui kai vienos įmonės darbuotojams, žmonių grupei, komandai, suteikiami iki 50 proc. mažesni tarifai, lengvatos pokalbiams ir t.t.

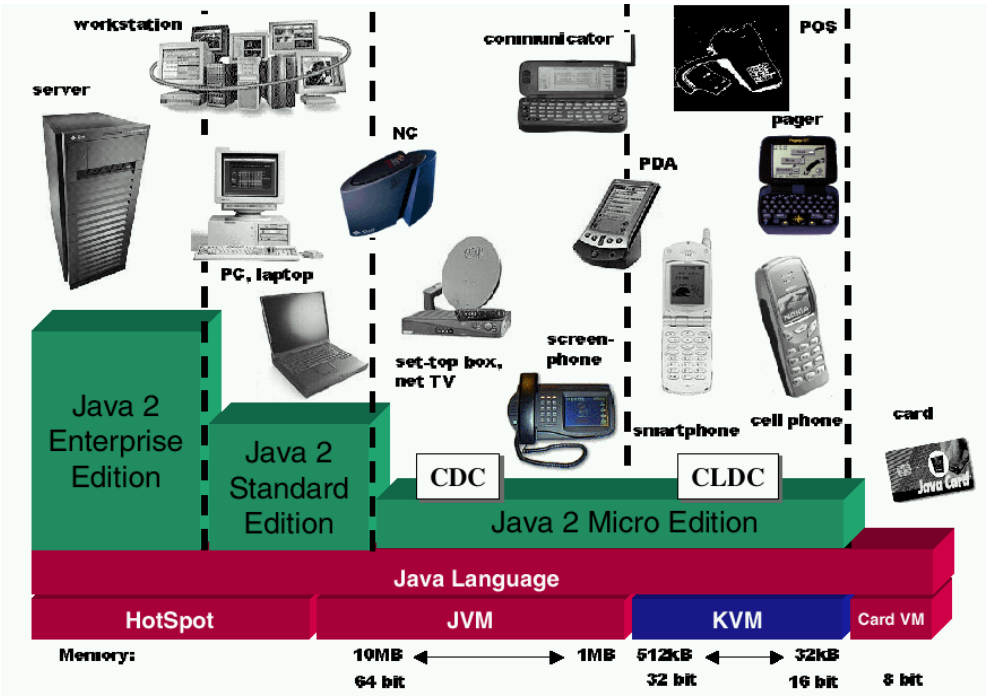
7. Minimalus plano mokestis – tai yra minimalaus mobilaus plano prakalbamas mokestis.

Kiekvienas mobiliojo mokėjimo planas turi ir daugiau parametru, tačiau buvo parinkti bendriausi iš jų atsižvelgiant į tai, kad šiuo metu žmogui išlieka aktualiausiais parametrais skambinimas ir trumpųjų pranešimų(sms) siuntimas.

Kai egzistuoja daugybė mobiliųjų planų pasiūlymų, darosi vis sunkiau išrinkti sau tinkamą, pagal konkrečius pageidavimus, planą, be to ne kiekvienas planas yra racionalus lyginant su kitais. Todėl sukurkime programą, kuri padėtų, situacijoje su mobiliaisiais mokėjimo planais, išrinkti reikiamą variantą, bet prieš tai trumpai apžvelgsiu kūrimo priemonę, kuri buvo pasirinktą šiai užduočiai įgyvendinti.

3. Java 2 Micro Edition

1999 metais vykusioje kasmetinėje "JavaOne" konferencijoje bendrovė "Sun Microsystems" pristatė nedideliems prietaisams pritaikytą "Java" versiją - "Java 2 Micro Edition" (J2ME). Iš esmės tai "apkarpytas" standartinės "Java" platformos ("Java 2 Standard Edition") variantas, sukurtas atsižvelgiant į visus mobiliųjų įrenginių apribojimus (nedidelę atmintį, ribotas procesorių galimybes, grafinės sąsajos ir informacijos įvedimo bei išvedimo ypatumus, nepastovų sujungimą su mobiliojo ryšio tinklu ir t. t.). Palyginti paprasta ir programinės įrangos kūrėjams laisvai prieinama belaidė J2ME technologija leidžia kurti galingas nedideles (vidutiniškai 40 KB) programas, kurios mobiliame įrenginyje (pavyzdžiui, telefone) padės įgyvendinti viską, kas tik programuotojui šauna į galvą - nuo paprastų žaidimų ir "gyvų" užsklandų iki sudėtingų programų darbui su tinklo serveriais. Abejonių nekelia ir naujos platformos pritaikymo galimybės. Belaidė "Java", naudojama mikroprocesorių kortelėse, pranešimų gavikliuose, mobiliuosiuose telefonuose, komunikatoriuose, automobilinėse telemetrijos sistemose ir kišeniniuose kompiuteriuose, užtikrina nepaprastą šių aparatų lankstumą ir padeda vartotojui veiksmingiau dirbti. Vos prieš keletą metų "mobilizacijos" keliu pasukusios "Sun Microsystems" atstovai neabejoja J2ME technologijos sėkme ir tikisi, jog programuotojai sugebės visapusiškai išnaudoti visas jos galimybes. O jos išties stulbinančios. "Motorola" tai įrodė dar 2001 metais, pasauliui pristačiusi su "Flashline" sukurtą "Java" programą, leidžiančią mobiliuoju telefonu valdyti robotą. "Lego Mindstorms" pavadintas robotas paklusniai vykdydavo komandas, siunčiamas iš "Motorola i50sx" telefono (suderinamo su J2ME). Jis galėjo ne tik vaikščioti, ploti rankomis, bet ir atlikti nesudėtingus šokio judesius[9].



6 pav. J2ME palyginimas su kitom JAVA platformom

Šiandien turint su J2ME suderinamą mobilųjį telefoną, galima naudotis populiaraus interneto pranešimų gaviklio ICQ funkcijomis (jeigu operatorius teikia tokią paslaugą), su "Bluetooth" pagalba valdyti "Winamp" grotuvo funkcijas, telefoną paversti į judesius reaguojančia apsaugos kamera (jei jame įmontuota skaitmeninė kamera) ir netgi identifikuoti save GSM tinkluose. Pastaroji savybė ypač naudinga, nes leidžia visapusiškai valdyti telefono funkcijas pagal telefono savininko buvimo vietą (darbas, namai, oro uostas, teatras, traukinių stotis, parduotuvė ir t. t). Kadangi išvardytos vietos atitinka konkrečias GSM tinklo ląsteles, vartotojas jas gali įtraukti į asmeninę duomenų bazę, o vėliau kiekvienai priskirti tam tikras funkcijas (pavyzdžiui, įspėjimo signalo aktyvavimą arba telefono režimų perjungimą patekus ar išėjus iš atitinkamos zonos). Atėjote į darbą - automatiškai įsijungia privatus režimas, vakare užsukote į teatrą - telefonas be įspėjimo "užmiega", grįžote namo - grąžinamas standartinis režimas ir panašiai. Tokia identifikacijos sistema nepakeičiama dažnai keliaujantiems: niekada nepramiegosite stoties, nes ją privažiavę išgirsite įspėjimo signalą (žinoma, jei minėta stotis užregistruota asmeninėje duomenų bazėje). Tai toli gražu ne viskas, ką gali šiuolaikiniai "Java" telefonai, tačiau nereiktų pamiršti, kad jų funkcionalumą lemia ne vien programinė įranga, bet ir konkretaus modelio techninės savybės.

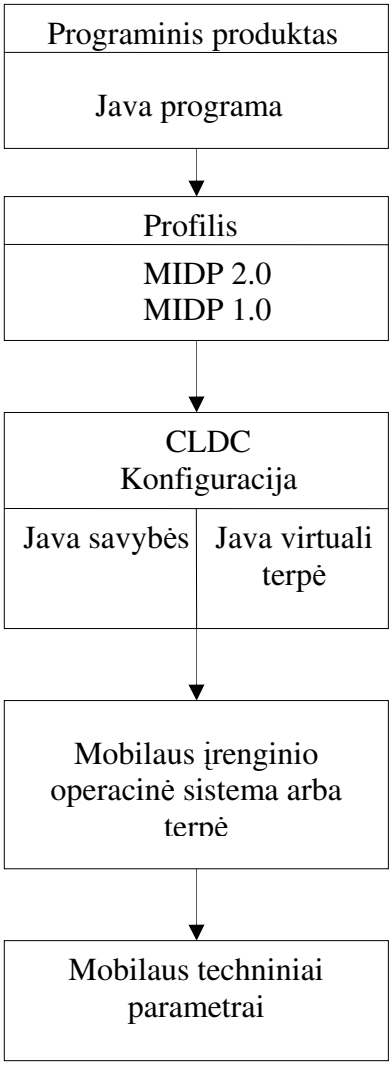
3.1 Ateities platforma

Palyginti su kitomis programavimo aplinkomis, "Java" platforma turi labai svarbų privalumą: ji beveik nepriklauso nuo programinės ir techninės elektroninių prietaisų įrangos. Vienintelė sąlyga - prietaisas turi būti suderinamas su virtualiąja "Java" mašina (Java Virtual Machine, JVM), kuri be "tarpininkų" vykdo visas "Java" kalba parašytas programas. Toks universalumas programuotojų pasaulyje ypač vertinamas. Bent šiek tiek susidūrusieji su "Java" kalba gerai žino, kad savo sintakse ji labai primena C++ programavimo kalbą. Tačiau yra vienas pagrindinis skirtumas. C++ programuotojas pirminį programos variantą visuomet priverstas priderinti atskirai kiekvienam įrenginiui, kuriame planuojama paleisti šią programą. Tuo tarpu "Java" programuotojui dėl to nereikia sukti galvos, nes visus "Java" kodus interpretuoja virtualioji "Java" mašina pačiame įrenginyje. Tai reiškia, kad tą pačią "Java" programą galima įdiegti į mobilųjį telefoną, kišeninį arba stalinį kompiuterį, taip pat į bet kurį įrenginį, kuriame yra JVM. Be atminties dydžio ir mobilaus įrenginio techninių savybių, vartotojas daugiau nesusidurs su jokiais apribojimais.

3.2 MIDP ir CLDC

Pagrindinė J2ME platformos dalis susideda iš dviejų konfigūracijų. Connected Device Configuration (CDC) ir Connected Limited Device Configuration (CLDC). Konfigūracija nustato Java technologijos naudojamas bibliotekas ir Java Virtual Machine(integruojama į mobilų įrenginį) galimybes. Kuo gi tos konfigūracijos skiriasi? CLDC konfigūracija susideda iš Java virtualios terpės, branduolio bibliotekų, klasių ir API (Application Programming Interface), kuri sukurta mobiliems įrenginiams su apribotais resursais(pvz. mobiliems telefonams). Tokiems, kurie turi 16 ir 32 bitų procesorius ir laisva atmintį iki 512 Kb. Egzistuoja dvi CLDC versijos 1.0 ir 1.1. CLDC 1.1 turi daugiau galimybių, pavyzdžiui viena iš jų tai plaukiojančio taško palaikymas[4].

CDC skirta hi-end tipo įrenginiams, tokiem kaip komunikatoriai, smartfonai, pagrinde su 32 bitų procesoriais ir laisva atmintimi apie 2 Mb.



7 pav. Mobilios Java programos sąveika su techniniais mobilaus parametrais

Daugelyje rinkoje mobiliųjų telefonų įdiegta "MIDP 1.0" (mobiliesiems prietaisams su CLDC pritaikyta pramoninio standarto specifikacija, Mobile Information Device Profile) pagrindu veikianti J2ME platforma. Kadangi programų dydį ir sudėtingumą riboja tik techninės įrenginių galimybės, programinės įrangos kūrėjai turi puikią progą pademonstruoti savo išradingumą. Tai suvokdami telefonų gamintojai taip pat nesnaudžia. Pavyzdžiui, darbui su "Nokia" telefonuose įdiegta "Java" technologija, jų galimybės gali keistis nuo dviejų programuojamų mygtukų iki QWERTY klaviatūros, nuo vienspalvio iki 4096 spalvų didelės skiriamosios gebos ekrano, nuo 150 KB atminties iki 16 ir daugiau MB - 92xx serijos komunikatoriuose, nuo standartinių jungčių iki naujosios "Pop-Port" jungties, užtikrinančios ne tik spartų duomenų perdavimą, bet ir sugebančios identifikuoti skaitmeninius priedus bei perduoti stereo garsą. Patobulinta mobiliųjų prietaisų specifikacija "MIDP 2.0" pasirodė 2003 metais ir todėl šiandienai dauguma mobiliųjų telefono gamintojų jau perėjo į šį standartą, kuriame atsirado daug naujų galimybių darbui su garsų, grafiką, padidintas saugumas, patogumas. MIDP užduodamos mobiliųjų įrenginių charakteristikos:

- ekrano skiriamoji geba minimum 96x54 pikseliai su ekrano taško gyliu minimum 1 bitas;
- įvedimo įrenginiu gali būti klaviatūra arba sensorių ekranas;
- 32 Kb dinaminės atmintis;
- 128 Kb MIDP komponentams;
- 8 Kb pastovių duomenų saugojimui;
- belaidys tinklas;
- maitinimas iš akumuliatoriaus.

Būtent specifikacija MIDP 2.0 ir CLDC 1.1 konfigūracija aš panaudojau kurdamas programą, kuri veiktų mobilioje aplinkoje. Nes būtent MIDP 2.0 leidžia padaryti programą patrauklesnę vartotojui, o CLDC 1.1 palaiko slankųjį tašką, be ko būtų žymiai sunkiau įgyvendinti skaičiavimus.

3.3 Java Virtual Machine

Java virtuali terpė, kuria naudoja CLDC konfigūracija, kiek skiriasi nuo standartinės Java naudojamos virtualios terpės. Virtuali Java terpė yra tiesiogiai integruotą į mobilųjį telefoną ir todėl kiekvienas mobilaus ryšio telefonų gamintojas turi atsižvelgti į priimtus standartus gamindamas mobilius įrenginius.

3.4 Saugi platforma

Galimybė savarankiškai kurti programas ir laisvai diegti bei paleisti jas "Java" telefonuose perša mintį apie neapdairiai paliktas saugumo spragas, kurios išradingų įsilaužėlių ir virusų kūrėjų rankose labai

greitai gali virsti klastingais spąstais telefono savininkui. Ar pagrįsta ši baimė, mėgino išsiaiškinti Vokietijos mobiliojo ryšio operatoriaus "T-Mobile" ekspertas Markusas Schmallas, nusprendęs paieškoti silpnųjų "Siemens SL45" vietų (telefonas suderinamas su J2ME platforma).

Kad telefonu negalėtų pasinaudoti nepageidaujami asmenys, J2ME platformoje iš anksto numatyti keli "saugikliai". Pirmiausia "apkarpytas" matematinių funkcijų rinkinys, apribotos programų paleidimo galimybės (viena programa negali paleisti kitos), neįmanomas tiesioginis priėjimas prie failų. Markuso Schmallo nuomone, viskas, ką gali padaryti piktavalis "Java" kodas, - gauti priėjimą prie informacijos kaupiklio, vidinės atminties, prisijungti prie interneto ir sąveikauti su telefone įdiegtomis programomis. Jis taip pat tvirtina, kad J2ME platforma visiškai apsaugo nuo virusų plitimo. Net ir tokie virusai kaip "Strange Brew" (pirmasis besidauginantis "Java" virusas) ar "Bean Hive" (turintis ypač gudriai paslėptą dauginimosi mechanizmą) mobiliesiems telefonams nekelia jokios grėsmės, nes vienas iš jų nesugeba daugintis (net ir asmeniniame kompiuteryje), o kitas - "Bean Hive" - naudoja su J2ME platforma nesuderinamas funkcijas.

Tiesa, testų metu su SMS žinutėse naudojamų (tik telefonams suprantamų) kodų pagalba pavyko "pakabinti" telefoną, tačiau dėl to buvo kalta ne "Java", o telefone naudojama gamintojo mikroprograma ("firmware"). Būtent dėl šios priežasties negalima vienareikšmiškai vertinti kitų gamintojų telefonų. Testų rezultatai rodo, kad kol kas nėra rimto pagrindo nerimauti, tačiau pradėjus masiškai naudoti patobulintą mobiliųjų prietaisų specifikaciją "MIDP 2.0", leidžiančią dirbti su failų sistema ir telefonų knyga, mobiliojo ryšio tinklais plintančio skaitmeninio užkrato rizika vis didėja. Beje, palyginti su "MIDP 1.0", naujoji versija turi platesnių darbo su garsu, vaizdu ir grafika galimybių, leidžia programose naudoti tonus ir WAV failus, užtikrina didesnį programų lankstumą keičiantis informacija su serveriais.

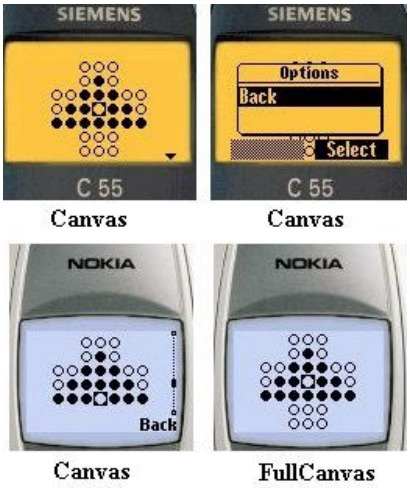
3.5 J2ME palyginimas su BREW platforma.

Kai belaidė "Java" jau gana aktyviai skynėsi kelią į mobiliųjų įrenginių rinką, pirmuosius žingsnius žengė ir tuo metų grėsmingiausias jos varžovas - korporacijos "Qualcomm" sukurta BREW (Binary Runtime Environment for Wireless™) platforma. Tai jungiamoji grandis tarp mobiliojo įrenginio (nebūtinai telefono) procesoriaus ir C/C++ kalba parašytų programų. Kaip ir "Java", BREW yra atviras standartas, suteikiantis programuotojams visišką kūrybinę laisvę. Pagrindiniai jo privalumai - mažesni kuriamų programų reikalavimai įrenginyje naudojamos atminties dydžiui, galimybė kurti sudėtingas programas darbui su dideliais duomenų masyvais (naudojantis 3G mobiliojo ryšio tinklais tai bus ypač aktualu) ir suderinamumas su "Macromedia Flash" formatu, automatiškai priderinančiu vaizdą prie įrenginyje naudojamo ekrano skiriamosios gebos.

2003 m. su BREW technologija susijusias programų siuntimo paslaugas tiekė apie 20 didelių mobiliojo ryšio operatorių visame pasaulyje, o ja besinaudojančiųjų skaičius 2002 metais siekė 3,2 milijono ("Java" tuo metu naudojosi jau daugiau nei 50 mln. vartotojų). Nepaisant tokio skirtumo, kiek netikėtas BREW populiarumas pakurstė tų metų aistras ne tik tarp telefonų gamintojų, bet ir mobiliųjų paslaugų rinkoje, kur dviem funkcionalumu viena kitai nenusileidžiančioms technologijoms būtų tikrai ankšta. Šiandienai kai įvyko techninės telefonų įrangos standartizacija "Java" turi didelį kozirį savo rankose. Sparčiai išpopuliarėjus daugialypės terpės telefonams ir komunikatoriams, turinčius standartinę ekrano skiriamąją gebą, jie jau gali visapusiškai išnaudoti visus J2ME privalumus. Todėl kol kas J2ME faktiškai neturi konkurentų. Lieka tik stebėti kas bus toliau.

3.6 J2ME priklausomybė nuo techninių įrenginio parametrų

Atsiradus dideliam mobiliųjų telefonų gamintojų skaičiui, kiekviena iš jų mato savo ateitį standartizuojant savo programinę įrangą. Todėl rinkoje egzistuoja daugybė programinės įrangos, skirtos, beveik kiekvienam, egzistuojančiam telefono modeliui. Mano manymu šioje sferoje pirmauja suomiška bendrovė „Nokia“, kuri diktuoja madas mobiliųjų telefonų rinkoje. Teoriškai kiekvienas programinis produktas, sukurtas J2ME pagalba, turėtų veikti pasirinktam mobiliam telefone[8]. Atsakykime į klausymą: kodėl tik teoriškai?



8 pav. Skirtumai tarp Siemens ir Nokia

Kiekvienas telefonų gamintojas neišvengiamai naudoja tik sau priimtinas konfigūracijas. Daugiausia tai susiję su telefono ekranu, bei grafinių interfeisu. Iliustruokime tai pavyzdžiu. „Siemens“ mobilieji telefonai, su spalvotu ekranu, turi skiriamąją geba 101x80, tuomet kai daugumą „Nokia“ spalvotų telefonų, turi skiriamąją geba 128x128 pikselių ir kiekvienas telefonas skirtingai vaizduoja vieną arba kitą

grafinį elementą. Išnagrinėkime paveikslą 8. Tarkime kai naudosime piešimą ekrane, objekto Canvas pagalba, tai Siemens piešimui skirs visą ekraną, komandas paslėpdamas apačioje trikampio pavidalų. Tuo tarpu Nokia neskiria piešimui viso ekrano ir komandas tiesiogiai parodo ekrano apačioje. Nokia yra galimybė naudoti objektą FullCanvas, kuris skiria piešimui visą ekraną, bet tuomet negalima įdėti jokiu komandų kas neturi prasmės. Todėl, kuriant programinį produktą pritaikytą konkrečiai mobiliojo telefono markei, reikėtų atsižvelgti į techninius mobiliojo parametrus.

3.7 Pagrindiniai J2ME privalumai

- lengvas diegti ir šalinti programas,
- galimybė kurti nedideles, bet funkcionalias programas,
  universalumas,
- neribotos pritaikymo galimybės,
- didžiulis nepriklausomų programinės įrangos kūrėjų susidomėjimas,
- nuolat didėjantis naudingų programų asortimentas,
- beveik visi naujausi telefonai turi įdiegtą J2ME technologiją.
- Įrenginiai su J2ME nedidelį ir patogus transportavimui

3.8 Mobilaus telefono emuliatoriai

Telefono emuliatorius – tai programa, leidžianti kompiuteryje paleisti programą, sukurtas mobiliam telefonui, atsižvelgiant į konkretaus telefono modelio techninius parametrus. Daugelis mobiliųjų telefono gamintojų išleidžia, telefono programavimui skirtas, priemones, kurios vadinamas SDK(Software Developer Kit).

Daugiausia pagalbinių programinių priemonių yra išleidusi suomių mobiliųjų telefonų bendrovė „Nokia“. Bendrovės puslapyje http://www.forum.nokia.com/main.html galima rasti daugybę „Nokia“ telefonams skirtos programinės įrangos bei emuliatorių.

Nokia padalino visus telefonų modelius į tam tikras platformų serijas(Series Developer Platform): 40, 60, 80 ir 90 serijos. Labai lengva telefonus į serijas galima būtų paskirstyti atsižvelgiant į konkretaus telefono modelio ekrano skiriamąją gebą:

- Serija 40 – šios serijos telefonų skiriamoji geba yra 96x65, 128x128, 128x160 pikselių;
- Serija 60 – šios serijos telefonų skiriamoji geba yra 176x208 pikselių;
- Serija 80 – šios serijos telefonų skiriamoji geba yra 640x200 pikselių;
- Serija 90 – šios serijos telefonų skiriamoji geba yra 640x320 pikselių;

Telefonai priklausantis serijai 40, neturi operacinės sistemos, todėl šiems, plačiai paplitusiems telefonams, programuoti galima tik J2ME programavimo kalboje, atsižvelgiant tik į tai ar palaiko konkretus telefono modelis Java.

Kiti 60, 80, 90 serijai priklausantis modeliai, turi Symbian operacinę sistemą ir leidžia programuotojui kurti programas ne tik su J2ME, bet ir su C++ programavimo kalba.

Mobiliųjų telefonų gamintoja „Siemens“ taip pat turi nemažą telefonų, palaikančiu Java technologijas pasirinkimą. Siemens Mobility Toolkit for Java Development tai pagrindinė priemonė, kuri leidžia programuotojams integruoti į ją konkretaus modelio telefono emuliatorių.

Čia galima būtų paminėti ir kitus mobiliųjų telefonų gamintojus tokius kaip „Motorola“, „Samsung“, „Sony Ericsson“, kurie taip pat pateikia įvairiausius emuliatorius bei pagalbines priemones, padedančias kurti bei eksploatuoti turimus mobilius telefonus.

9 paveiksle iliustruoti standartinis J2ME, 40 serijos „Nokia 6230“, bei Sony Ericsson K750 modelio emuliatoriai.



9 pav. standartinis J2ME, Nokia 6230, Sony Ericsson emuliatoriai.

3.9 Kodėl būtent J2ME?

Rašant Best Mobile Plans programą, J2ME programavimo kalba, dėl šių priežasčių:

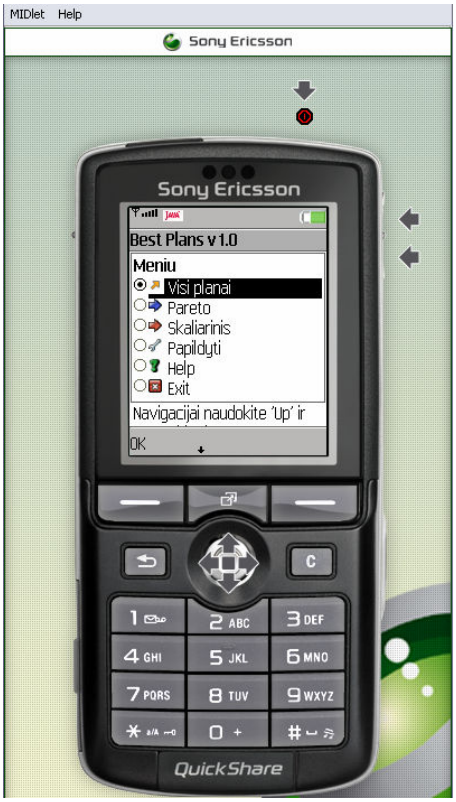
- Patogumas – kiekvienas naujesnio mobiliojo telefono, palaikančio Java vartotojas, gali turėti programą savo telefone.
- Lengvas diegimas, bei šalinimas.
- Neribotas pritaikymas.
- Suderinamumas su visom Operacinėm sistemom.
- Didėjantis populiarumas – atlikus apklausa http://www.juga.ru/ puslapyje ir uždavus klausymą „Kokia Java pritaikymo sritis jūs labiausiai dominą?“ J2ME surinko 25 % iš 444 lankytojų balsų.

4. Best Mobile Plans J2ME programa

Išnagrinėję vektorinės optimizacijos apibrėžimus bei metodus ir J2ME mobiliąją aplinką, sukursime programinę priemonę, kuri padėtų išspręsti, 2.7 skyriuje apžvelgtą pavyzdį su mobiliaisiais planais, kuri leistų, mobiliajam telefone, dviem vektorinės optimizacijos metodais, iš esamų planų sąrašo išrinkti sau tinkamiausia(10 pav.). Užduoties įgyvenimui pasirinkome du vektorinio optimizavimo metodus:

- **Skaliarinis** – kitaip pavadintas vektorinis optimizavimas su svoriai.(2.3 skyrius)

- **Pareto optimum** – leidžiantis atmesti visus neracionalius variantus, iš pateiktų parametrų sąrašo(2.2 skyrius)



10 pav. J2ME „Best Mobile Plans“ sukurtos programos iliustracija Sony Ericsson emuliatoriuje.

4.1 Panaudota programinė įranga ir reikalavimai

Programinę įranga, naudotą kuriant programą:

- JavaTM 2 SDK, Standard Edition – Java bibliotekos.
  Version 1.4.2

- J2ME Wireless Toolkit 2.2 – Mobilioji Java

- „NetBeans 4.0“ , bei integruota „NetBeans Mobility Pack“ papildymą, tam kad būtų galima kurti programas mobiliesiems telefonams.

Naudoti telefonų emuliatoriai:

- J2ME
- Nokia
- Sony Ericsson
- Motorola
- Siemens

Sisteminiai reikalavimai:

Miminalus „Java™ 2 SDK, Standard Edition“ sisteminiai reikalavimai:

- Solaris operacinė sistema:
  - Procesorius: 32 – bit arba 64-bit SPARC
- Linux operacinė sistema:
  - Procesorius: 450 MHz
  - Operacinė atmintis: 32 megabytes of RAM
  - Laisvos vietos diske: 75 megabytes

- Microsoft Windows operacinė sistema:
  - Procesorius: 500 MHz Intel Pentium III
  - Operacinė atmintis: 32 megabytes of RAM
  - Laisvos vietos diske: 75 megabytes

Miminalus „NetBeans 4.0“ sisteminiai reikalavimai:

- SolarisTM operacinė sistema:
  - Procesorius: 450 MHz UltraTM 10
  - Operacinė atmintis: 384 megabytes
  - Laisvos vietos diske: 125 megabytes

- Microsoft Windows operacinė sistema:
  - Procesorius: 500 MHz Intel Pentium III

- Operacinė atmintis: 384 megabytes
- Laisvos vietos diske: 125 megabytes

- Linux operacinė sistema:
  - Procesorius: 500 MHZ Intel Pentium III
  - Operacinė atmintis: 384 megabytes
  - Disk space: 125 megabytes

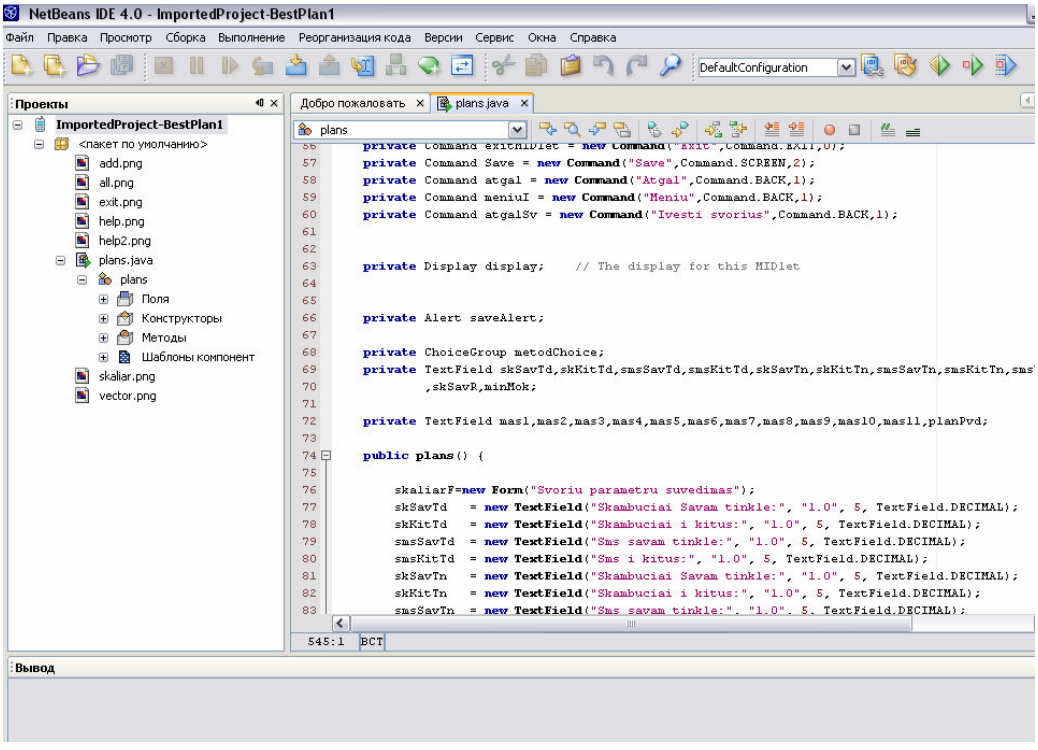
- Macintosh OS X operacinė sistema:
  - Procesorius: PowerPC G4
  - Operacinė atmintis: 512 megabytes
  - Laisvos vietos diske: 125

Reikalavimai mobiliajam telefonui:

- Java palaikymas
- MIDP 2.0 profilis
- CLDC 1.1 konfigūracija

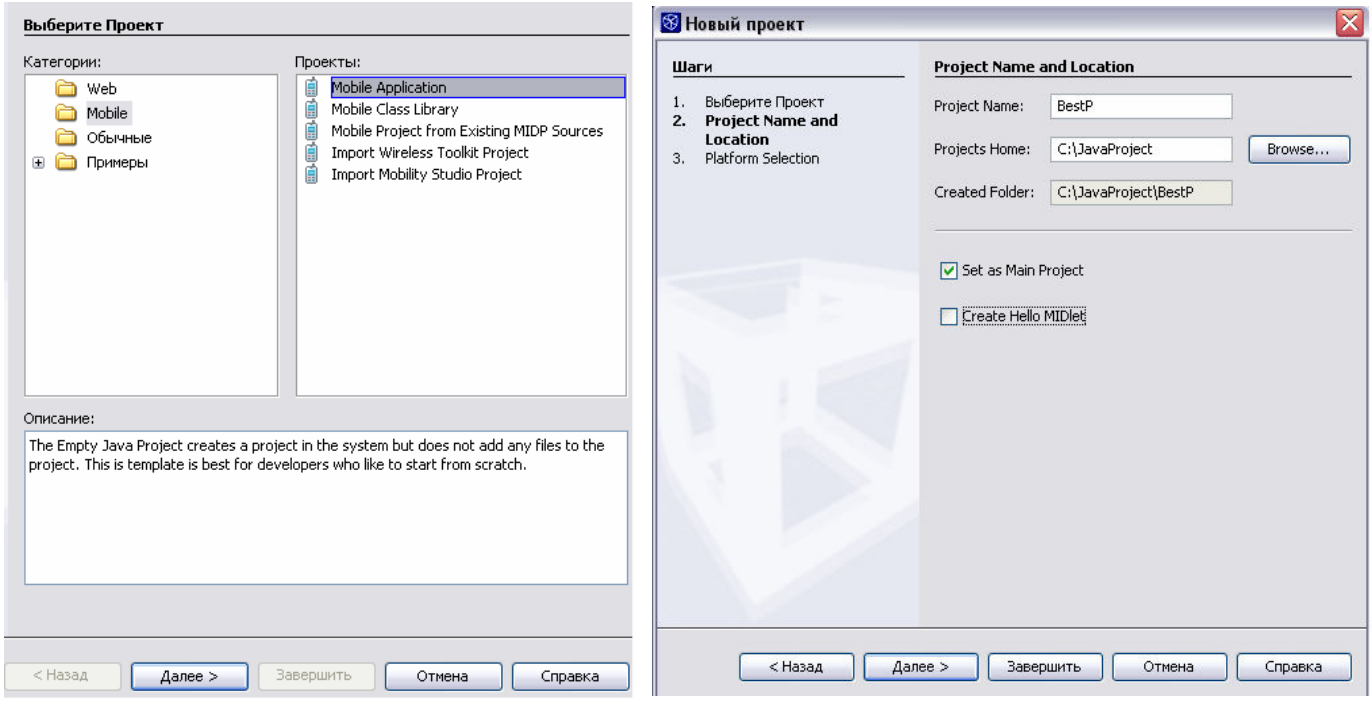
4.3 Programos Best Mobile Plans kūrimo aplinka

Pavaizduosime programos kūrimo „NetBeans“ aplinka:



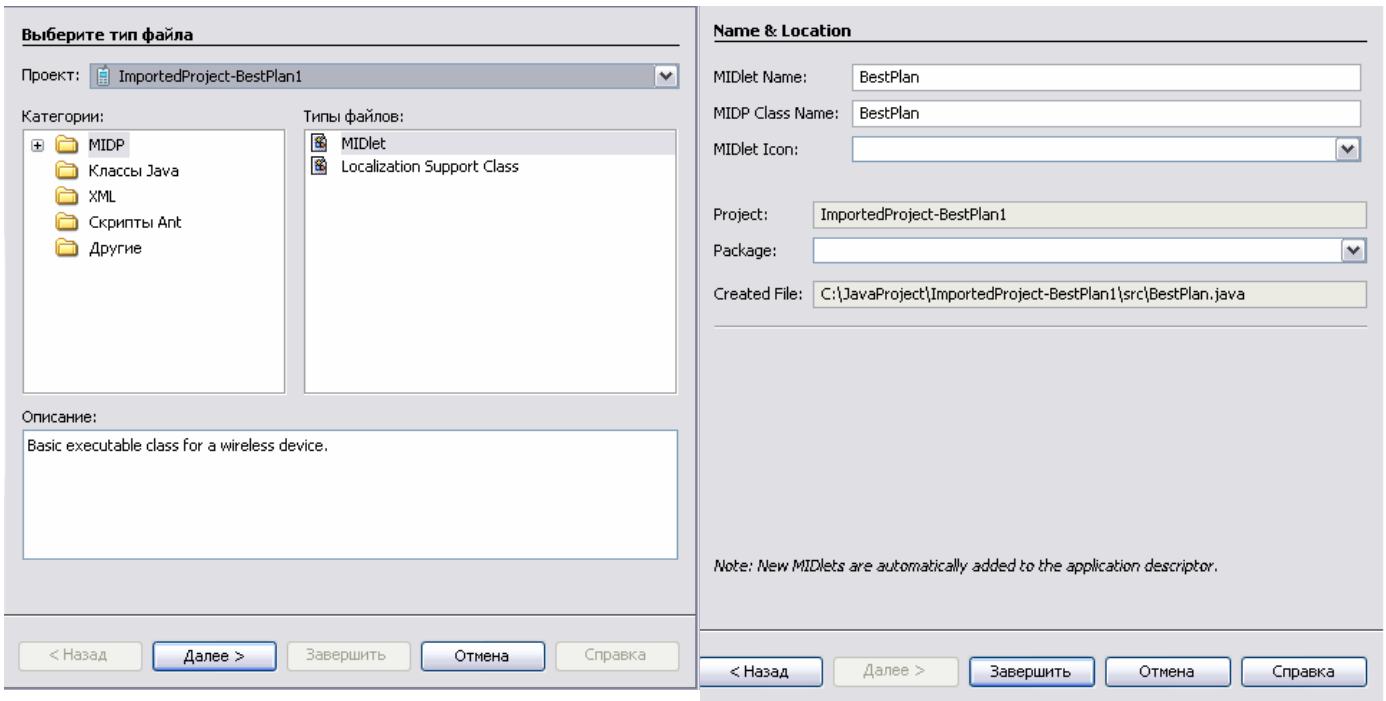
11 pav. NetBeans 4.0 aplinka

Pradiniame žingsnyje sukuriamas mobilus projektas:



12 pav. Mobilaus projekto kūrimas

Sukūrus projektą įkeliam į jį failą „BestPlans.java“(13 pav.):



13 pav. Failo įkėlimas į projektą

Prieš paleidžiant programą, pasirinkus projekto „Properties“, galima jį suderinti su konkretaus modelio mobiliojo telefono emuliatoriumi. Mūsų atveju buvo naudotą standartinė konfigūracija, naudojant J2ME standartinį „DefaultColorPhone“ emuliatorių, kuris yra integruotas į „J2ME Wireless Toolkit 2.2“ paketą.

4.4 Programoje panaudoti J2ME paketai

Kuriant „BestPlans“ programą buvo panaudoti šie Java 2 ME profilio MIDP 2.0 ir CLDC 1.1 konfigūracijos paketai:

- java.lang
- javax.microedition.lcdui
- javax.microedition.midlet

- Java.lang – paketas turi savyje sistemines klases, Java kalbos pagrindus, bei išimtis.

- Java.microedition.lcdui – paketas turi savyje įvairias klases, programos interfeiso mobilioje aplinkoje realizavimui. Egzistuoja daugybė klasių, tikrai gražių ir ispūdingų J2ME programų mobiliam sukūrimui.

- Java.microedition.midlet – labai nedidelis paketas, bet kuris turi labai didelę svarbą mobiliųjų programų kūrime. Su šio paketo pagalba įvyksta sąveika tarp sukurtos programos ir mobiliuoju telefonu profilio MIDP pagalba.

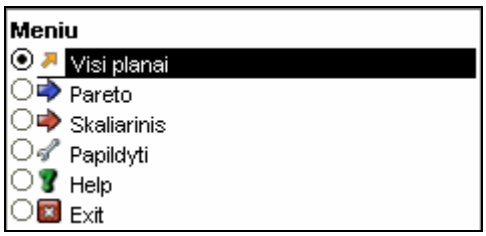
4.4 Programos aprašas

Paleidus programą, „NetBeans“ aplinkoje mygtuko  arba pasirinkto emuliatoriaus, pagalba pamatysime pagrindinį programos langą(14 pav.):



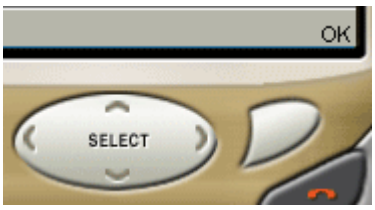
14 pav. Pagrindinis „BestPlans“ programos langas

Atsidarius emuliatoriaus langui matome pasirenkama programą „BestPlans“. Tam kad ją paleisti spaudžiame mygtuką „Launch“ (14 pav.). Paleidus „BestPlans“ telefono ekrane pamatysime pasirinkimo „Meniu“(15 pav.):



15 pav. Pasirinkimo meniu

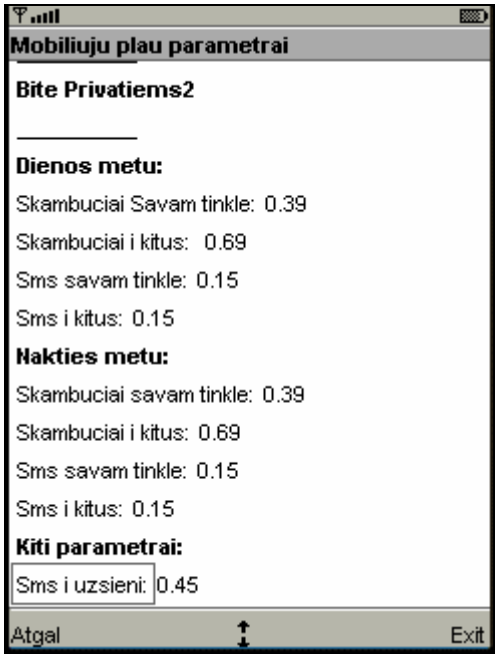
Su emuliatoriaus navigacijos mygtukų pagalbą (16 pav.) galime parinkti norimą „Meniu“ punktą, ir patvirtinus klavišų „Select“, galime paspaudus „OK“ pereiti į pasirinktą punktą:



16 pav. Navigacijos bei patvirtinimo mygtukai

Meniu:

- ***Visi planai***(17 pav.) – parinkus šį punktą telefonas atvaizduos informaciją apie turimus mobiliuosius planus(2.7 skyrius), kuriu informacija galime peržiūrėti navigacijos mygtukų pagalba. Čia vartotojas gali susipažinti su įvestais parametrais bei jų reikšmėmis.



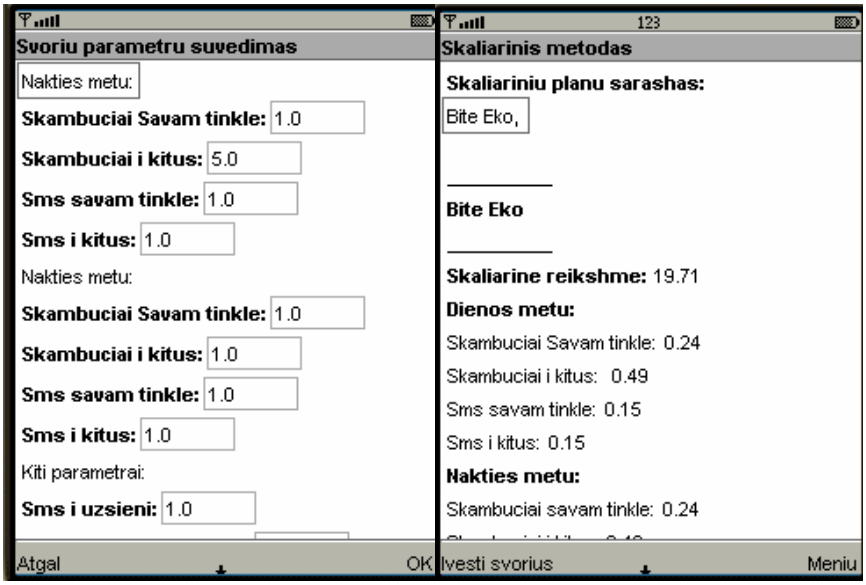
17 pav. Meniu punktas visi planai

- ***Pareto*** - parinkus šį punktą iš visų planų bus atmesti neracionalus planai, o atvaizduoti tik planai kurie atrinkti pagal Pareto Optimalumą, pagal (2.1) bei (2.2) formules(2.2 skyrius). Vartotojui bus pateikti tik atrinkti planai(18 pav.) ir todėl jis jau galės iš šių planų atsirinkti sau tinkamą ir patogesnį. Telefone bus atvaizduotas išrinktų planų sąrašas, o žemiau, navigacijos mygtukų pagalba, galima peržiūrėti šių mobilaus mokėjimo planų parametrus. Naudotas algoritmas 1 priede.



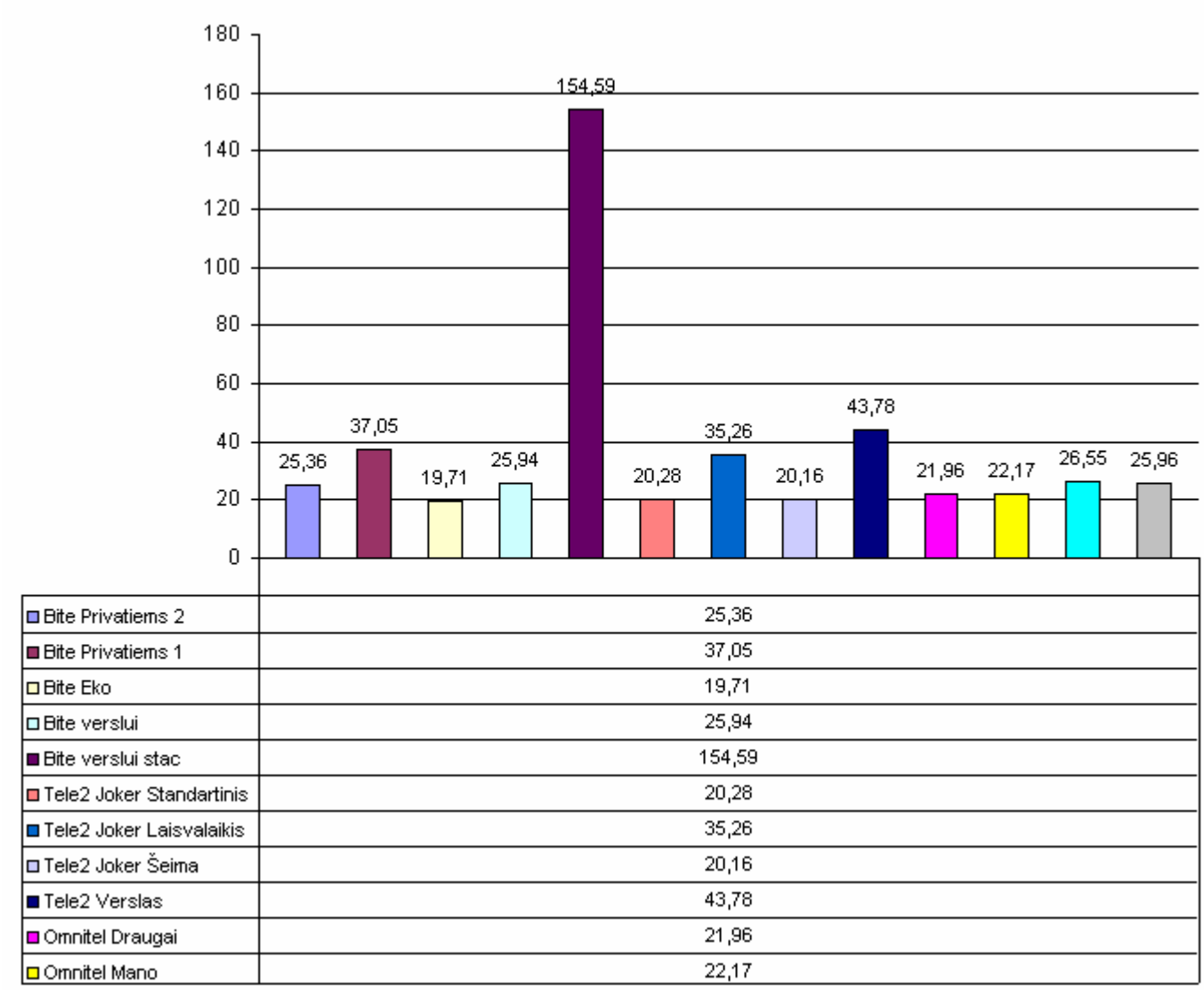
18 pav. Pareto optimalus mokėjimo planai

- ***Skaliarinis*** - parinkus šį punktą, emuliatoriaus ekrane bus atvaizduota svorių įvedimo forma(18 pav.), kurioje kiekvienam mobiliojo mokėjimo plano parametrui, galima įvesti svorį(Skyrius 2.3). Apibudinkime plačiau, kas tai yra svoris? Taip galima pavadinti kiekvieno parametro svarbumą atskiram vartotojui. Pavyzdžiui, tarkime vartotojas pastoviai skambina į kitus tinklus, o sms(trumpųjų pranešimų) beveik visai nerašo, todėl jam parametro „Skambučiai į kitus tinklus“ svarbumas yra didesnis negu parametro„sms“. Reiškia toks vartotojas turėtų parametrui „Skambučiai į kitus tinklus“ užduoti didesnį svorį, negu pavyzdžiui parametrui „Sms i kitus tinklus“. Po to paspaudus mygtuką „OK“ pagal formulę (2.3) bus išrinktas planas atitinkantis vartotojo pageidavimus(18 pav.).



18 pav. Svorių įvedimo forma, bei rezultatų atvaizdavimas

Kiekvieną kartą suvedus svorius, pagal formule (2.3) apskaičiuojamos skaliarinės reikšmės kiekvienam planui, sprendiniu bus mažiausia reikšmė. Atskiru atvejų gali būti išrinkti ir kelį planai, bet tai atsitiks tik tuo atveju kai skaliarinės reikšmės sutaps. Atvaizduokime tos reikšmės kitimą kiekvienam planui, jeigu kiekvienam parametrui užduosime svori 1.0, o parametrui „Skambučiai į kitus tinklus“ svorį 5.0 (19 pav.):

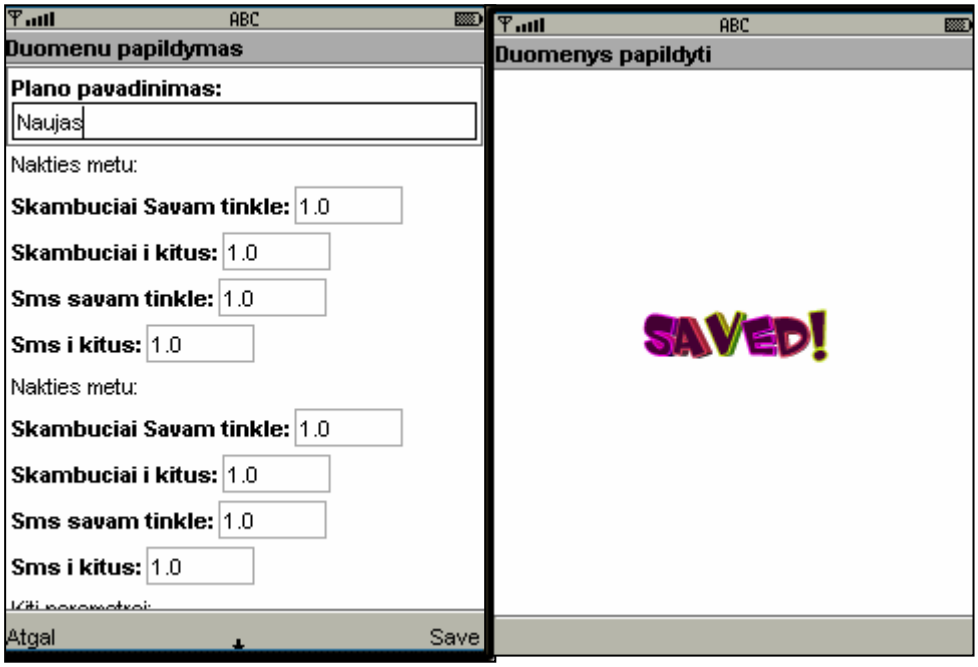


Bite Privatiems 2	25,36
Bite Privatiems 1	37,05
Bite Eko	19,71
Bite verslui	25,94
Bite verslui stac	154,59
Tele2 Joker Standartinis	20,28
Tele2 Joker Laisvalaikis	35,26
Tele2 Joker Šeima	20,16
Tele2 Verslas	43,78
Omnitel Draugai	21,96
Omnitel Mano	22,17

19 pav. Skaliarinės reikšmės kitimo grafikas

Iš grafiko galime matome, kad minimali reikšmė yra 19,71 ir pagal šios svorius vartotojui geriausiai tinkamas mokėjimo planas Bite Eko ir programą išrinko kaip tik jį(18 pav.). Algoritmas 2 priede.

- ***Papildyti***(20 pav.) – Šį mobiliajam telefonui skirta programa, turi mobiliųjų planų papildymo galimybę. Parinkus šį menių punktą bus atvaizduota forma duomenų papildymui. Papildžius duomenys spaudžiame mygtuką „Save“.



20 pav. Duomenų papildymo forma.

- ***Help*** – Parinkus šį menių punktą bus atvaizduota trumpa pagalba vartotojui.
- ***Exit*** – Parinkus šį menių punktą bus baigtas programos darbas.

4.5 Programos testavimas mobiliose aplinkose

Išnagrinėję programos struktūra labai naudinga būtų palyginti jos veiklą įvairiuose telefonų modeliuose. Programai testuoti buvo pasirinkti šie telefonų modeliai:

- Motorola V360
- Nokia 6230
- Sony Ericsson K750
- Siemens SK65

Visi pateikti telefonų modeliai palaiko MIDP 2.0 ir CLDC 1.1. Pateikime keletą kiekvieno telefono parametrų:

- Motorola V360
  - Ekrano skiriamoji geba: 176x220

- Spalvos: 18 bitų (262,144 spalvų)
- Maksimalus įrašomo JAR failo dydis: 100 Kb
- Saugomos atminties dydis: 1.8 Mb
- Operatyvios atminties: 800 Kb
- Palaikomos technologijos: Bluetooth technology API(Application program interface), Wireless Messaging API 2.0, Mobile Media API.

- Nokia 6230
  - Ekrano skiriamoji geba: 128x128, o piešimui skirta tik 128x96
  - Spalvos: 16 bitų (65,536 spalvų)
  - Maksimalus įrašomo JAR failo dydis: 128 Kb
  - Saugomos atminties dydis: 3,5 Mb
  - Operatyvios atminties: 512 Kb
  - Palaikomos technologijos: Bluetooth technology API(Application program interface), Wireless Messaging API, Nokia UI API, Mobile Media API 1.1, Java Technology Wireless Industry

- Sony Ericsson K750
  - Ekrano skiriamoji geba: 176x220
  - Spalvos: 18 bitų (262,144 spalvų)
  - Maksimalus įrašomo JAR failo dydis: Neapribotas(priklausomai nuo laisvos atminties)
  - Saugomos atminties dydis: 38 Mb
  - Operatyvios atminties: 1.5 Mb
  - Palaikomos technologijos: Java Technology Wireless Industry, Mobile Media API, Wireless Messaging API, PDA API, : Bluetooth technology API i, 3D API

- Siemens SK65
  - Ekrano skiriamoji geba: 132x176
  - Spalvos: 16 bitų (65,536 spalvų)
  - Maksimalus įrašomo JAR failo dydis: 350 kb
  - Saugomos atminties dydis: 11 Mb
  - Operatyvios atminties: 5 Mb
  - Palaikomos technologijos: Wireless Messaging API, Mobile Media API, Java Technology Wireless Industry, 3D API, location API, siemens-color-game, siemens-extension-API, Bluetooth technology API

Kas liečia programinę įrangą skirta telefonams, ją galima atsisiusti iš kiekvieno telefono gamintojo oficialių puslapių. Buvo panaudota šį:

- Motorola Launchpad for j2ME
- Nokia Developer‘s Suite 2.0 for j2ME, Series 40 Developer Platform 2.0 SDK
- Sony Ericsson j2ME SDK 2.2
- Siemens Mobility Toolkit bei SK65 emuliatorius

Paleidę sukurta programa mygtuku „ Launch“ kiekviename emuliatoriuje išvydome(21 pav.):



21 pav. Meniu kiekviename iš telefono modelių

Kaip matosi iš 21 paveikslo vienintelis Sony Ericsson atvaizdavo meniu esančius paveiksliukus, ko kiti telefonai neparodė. Kas liečia Nokia telefoną tai čia buvo paslėpta OK komanda, kas privedė prie to, kad pradžioje reikėjo įeiti į "options" meniu ir tik po to paspausti OK. Tai yra gana nepatogu vartotojo atžvilgiu. Motorola ir Siemens čia pasirodė gana apilygiai, nors mano manymu Siemens ekrane meniu atrodo patraukliau negu Motoroloje. Šioje situacijoje galima išskirti du lyderius. Besąlygiškai pirmas būtų Sony Ericsson K750, o šalia jo Siemens SK65 bei Motorola. Kas liečia navigacijos mygtukus patogiausi pasirodė Sony Ericsson telefone, bet tai daugiau skonio reikalas.

Pabandykime pasirinkti menių punktą „Visi planai“ (22 pav.):



22 pav. Meniu punkto „Visi planai“ atvaizdavimas kiekviename iš telefono modelių

Akivaizdžiai matosi(22 pav.), kad kiekvienas telefonas skirtingai ir savaip atvaizdavo mobiliųjų mokėjimo planų informacija. Galima pastebėti, kad Motorola gana neblogai atliko šią užduotį. Jos emuliatorius atvaizdavo duomenys taisyklingai ir jų daugiausia tilpo ekrane. Lyginant Sony Ericsson su Motorola, kurių ekrano skiriamoji geba vienoda 176x220, matome, kad Sony Ericsson atvaizdavo mažiau informacijos. Tai galima paaiškinti tuo, kad Sony Ericsson skiria ne visą ekraną piešimui, nors parametruose nurodyta kitaip. Nepaisant šio pastebėjimo visgi Sony Ericsson atvaizdavo, nors ir ne tiek duomenų kiek Motorola, bet jos atvaizdavo taisyklingai. Imant mobilius emuliatorius „Nokia“ bei „Siemens“ pastebėsiu, kad ne visai tikslus duomenų atvaizdavimas susijęs su ekrano skiriamąją gebą. Nokia skiriamoji geba 128x128, o Siemens 132x176. Be to Nokia skiria ekrano piešimui tik 128x96 šios gebos. Čia galime padaryti išvadą, kad nepaisant didesnės ekrano skiriamosios gebos Siemens žymiai blogiau susidorojo su užduotimi, ekrane atvaizduodamas neaiškiai išsibarsčiusius simbolius. Galima pastebėti, kad programoje aprašytą „\n“ simbolį, kuris perneša žymeklį į kitą eilutę, Siemens emuliatorius nepastebėjo.

Panagrinėkime kaip mobiliuosius skaičiavimus. Greičiausiai, pagal pareto optimum, mobiliuosius mokėjimo planus atrinko Motorola ir Sony Ericsson, labai nedaug atsiliko Nokia. Kas liečia Siemens jis palyginus su kitais skaičiavo apilygiai 5 sekundėm daugiau, nors tai gali būtį susiję su Siemens programinę įranga, bei emuliatorium. Vektorinį skaičiavimą su svoriais, uždavus kiekvienam parametrui svorį 20, visi telefonai skaičiavimus atliko apilygiai.

Apibendrinant aukščiau paminėtus paliginimus, bei telefonų modelius galimą būtų išskirti du telefonus: Motorola bei Sony Ericsson. Kurie gražiai atvaizdavo visą programą, nors jį buvo kurta standartiniam j2ME emuliatoriui. Galima pastebėti, kad vienintelis Sony Ericsson atvaizdavo programoje paveiksliukus(piktogramas). Taip pat maloniai nustebino Sony Ericsson programinė įranga. Matosi, kad šį bendrovė aktyviai bendradarbiauja su Sun Microsystem(j2ME kūrėja) ir todėl programinės priemonės atrodo labai panašiai(23 pav.) kaip ir teikiamo j2ME emuliatoriaus. Skiriasi tik emuliatoriai.



23 pav. Pradinis langas pasirodantis startuojant Sony Ericsson programinei priemonei

Todėl, kuriant j2ME programą Sony Ericsson telefonui, galima testuoti standartiniame emuliatoriuje. Kas liečia Nokia, bei Siemens, kurie pasirodė prasčiau, galima pažymėti tai, kad tai gali būti susiję su mažesne skiriamąją geba bei programinę įrangą, kuria teikia šių telefonų gamintojai.

Reziumuojant šį palyginimą pastebėsiu, kad telefonų emuliatoriai, yra labai naudinga priemonė. Nes įsivaizduokite kiek reikėtų pinigų programuotojui išleisti kiekvieną kartą perkant telefoną naujos programos testavimui. Lieka pastebėti, kad kuriant programą, reikia būtinai atsižvelgti į konkretaus

telefono modelį, bei jo parametrus. Nes, testuojant programa standartiniame „DefaultColorPhone“ j2ME emuliatoriuje, programuotojas gali susidurti su sunkumais kai reikės programą paleisti konkretaus modelio telefone.

5. Išvados ir Pasiūlymai

Apžvelgus vektorinio optimizavimo metodus galima pastebėti, kad daugelis iš jų lengvai pritaikomi ir įgyvendinami kasdieniame gyvenime. Svarbu tai, kad pritaikyti optimizavimo uždavinius galima daugelyje prekybos, verslo ir kitų sričių. Tai gana svarbu, nes kaip buvo paminėta aukščiau, konkuruoti tenka su tais kurie optimizuoja.

Buvo sukurta programą mobilioje J2ME aplinkoje, kuri, dviem vektorinio optimizavimo metodais, atrenka iš pateiktų mobiliųjų mokėjimo planų sąrašo optimaliausius planus. Atsižvelgiant į programos pateiktus rezultatus galima pasakyti kad parinkta mobiliųjų mokėjimo planų sritys pasirodė tinkama šiai užduočiai. Galiu pastebėti, kad šią programą galima lengvai pritaikyti ir kitoms sritims, pakeitus joje esančius duomenys bei parametrus.

Uždaviniui pasirinktos priemonės (*Java 2 Micro Edition* bei NetBeans kūrimo priemonė) pasirodė tinkamos ir labai patogios šiam uždaviniui sukurti.

Programą buvo kurtą pasitelkiant testavimui standartinį „DefaultColorPhone“ Java 2 Wireless Toolkit emuliatorių ir teoriškai jį turėtų gerai veikti kiekviename telefono modelyje, tačiau išanalizavus programos veikimą kitose telefonų emuliatoriuose, kai kur būtą grafinių neatitikimų. Nepaisant to, optimalus mobilieji mokėjimo planai, kiekviename emuliatoriuje, buvo išrinkti teisingai. Grafinius neatitikimus galima paaiškinti tuo, kad kuriant programą, reikėtų ją kurti konkrečiam telefono modeliui, atsižvelgiant į jo techninius parametrus.

Išnagrinėjus sukurtą programą, galima surasi keletą funkcijų, kurias reikėtų joje patobulinti. Visų pirma, galima pagerinti vartotojo sąsają, padaryti ją patogesne. Vartotojų patogumui būtų galima įgyvendinti daugiau funkcijų, susijusių su duomenų redagavimu.

6. Literatūros sąrašas

1. A. Žilinskas. Matematinis programavimas. Kaunas: Vytauto Didžiojo universiteto leidykla, 2000, 227 p.
2. А. Жилинскас, В. Шалтянис. Поиск оптимума. Москва: Наука, 1989, 128 с.
3. A. Žilinskas. Naujieji projektavimo metodai. Vilnius: Mokslas, 1990, 109 p.
4. С.Г. Горнаков. Программирование мобильных телефонов на Java 2 Micro Edition. Москва: ДМК Пресс, 2004, 336 с.
5. Martin J. Wells. J2ME ™ Game Programming. Premier Press, 2004, 803 p.
6. Habil. dr. prof. Jonas Mockaus asmeninis interneto puslapis, skirtas optimizicijai. http://soften.ktu.lt/~mockus
7. Sun Microsystems Java 2 Wireless toolkit. http://java.sun.com/products/j2mewtoolkit/
8. Сайт потребителей и разработчиков Java. http://www.juga.ru/
9. Naujoji komunikacija. http://www.nktv.lt/
10. Сайт потребителей и разработчиков Java. http://www.javable.com/
11. NetBeans. http://www.netbeans.org/kb/41/mobility.html
12. Omnitel. http://www.omnitel.lt/
13. Tele2. http://www.tele2.lt/
14. Bitė. http://www.bite.lt/
15. Sony Ericsson. http://www.sonyericsson.com/
16. Nokia Developers. http://www.forum.nokia.com/
17. Siemens Developers. http://www.siemens-mobile.com/developer
18. Motorola. http://www.motocoders.com/motorola/pcsHome.jsp

7. Priedai

1 PRIEDAS

Neracionaliu mobiliųjų mokėjimo planu atmetimas pagal Pareto Optimum

```
public void Pareto(){

        Form formVector=new Form("Pareto mobilieji planai");

        int i=0,e=0,k=0,l=0, nr=0,pr=0,pab=parSk,planuSkTmp=planuSk,temp=0;
        boolean geras=true, param_geriau=false,param_blogiau=false;
        int planDisplayIndex[]=new int[planuSkTmp];

        for(e=0;e<planuSkTmp;e++){
           planDisplayIndex[e]=e;
        }

        while(nr==parSk)
           nr++;

        while (nr<parSk){
           pr=pab=nr;
           while (nr+1<parSk){
              nr++;
              pab=nr;
           }

           // eliminuoja blogus planus
           for ( e = planuSkTmp-1; e >= 0; e-- ) {
              geras = true;
              for ( k = 0; (geras == true) && (k < planuSkTmp); k++) {
                 if ( k == e ) // netikrina plano su savimi
                    continue;
                 // tikrina ar planas i geras lyginant su k
                 param_geriau = false;
                 param_blogiau = false;
                 for ( l = pr; (param_geriau == false) && (l <= pab); l++) {
                    if ( PlansData[planDisplayIndex[e]][l] > PlansData[planDisplayIndex[k]][l] )
                       param_blogiau = true; // kuo mazesnis, tuo geresnis
                    if ( PlansData[planDisplayIndex[e]][l] < PlansData[planDisplayIndex[k]][l])
                       param_geriau = true;
                 }
```

```
                // Jeigu planas e blogiau uz k
                if ( !param_geriau && param_blogiau)
                    geras = false;
              }
              // ishmetam e jei blogas
              if (!geras) {
                for (k=e; k < planuSkTmp-1; k++)
                    planDisplayIndex[k]=planDisplayIndex[k+1];
                planuSkTmp--;
            }
          }
          nr++;
      }
  }
}
```

BestPlans.java

Geriausio mobilaus mokėjimo plano suradimas įvedant svorius

```
public void Skaliar(){
      Form skaliarS=new Form("Skaliarinis metodas");
      double skaliar[]=new double[planuSk];
      int e,s;
      for(e=0;e<planuSk;e++){
         skaliar[e]=0;
         for(s=0;s<parSk;s++){
            skaliar[e]+= PlansData[e][s]*svorList[s];
         }
         //skaliarS.append(String.valueOf(skaliar[e]));
      }

      //randam geriausia
      double geriaus = skaliar[0];
      for(e=0;e<planuSk;e++){
         if(skaliar[e]<geriaus){
            geriaus=skaliar[e];
         }
      }


      skaliarS.append(new StringItem("Skaliariniu planu sarashas:",""));
      skaliarS.append("\n");
      for(e=0;e<planuSk;e++){
         skaliarS.append(String.valueOf(skaliar[e]));
         if(geriaus==skaliar[e]){
            skaliarS.append(planai[e]+", ");
            skaliarS.append("\n");
         }
      }


      for (e=0; e<planuSk;e++){
         if(geriaus==skaliar[e]){
            .....
            išvedimas
         }
      }


      skaliarS.addCommand(atgalSv);
      skaliarS.addCommand(meniuI);
      skaliarS.setCommandListener(this);
      display.getDisplay(this).setCurrent(skaliarS);
}
```

BestPlans.java