KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Arvydas Matulevičius

OPTIMIZACINIŲ METODŲ TYRIMAS MOBILIOJO RYŠIO APLINKOJE

Magistro darbas

Darbo vadovas Prof. J. Mockus

Kaunas, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Arvydas Matulevičius

OPTIMIZACINIŲ METODŲ TYRIMAS MOBILIOJO RYŠIO APLINKOJE

Magistro darbas

Kalbos konsultantė		Vadovas	
	Lietuvių k. katedros lekt.		Prof. J.Mockus
	I. Mickienė		2006-05-29
2006-05-29			
Recenzentas		Atliko	
	Docentas		IFM-1gr.studentas
	D. Rubliauskas		Arvydas Matulevičius
2006-05-29		2006-05-29	

Kaunas, 2006

Turinys

1. Įvadas

Ši tema buvo pasirinkta todėl, kad šiuo metu rinkoje įsivyrauja mobilūs prietaisai (tokie kaip mobilūs telefonai, kišeniniai kompiuteriai) palaikantys naujas technologijas. Šių įrenginių paklausa ir pasiūla nuolat didėja.Vieni vartotojai įrenginius renkasi pagal dizainą, kiti gi atsižvelgia į telefono funkcijas, programinę įrangą, bei galimybę naudotis . Daugelyje naujų telefonų įdiegta J2ME (Java 2 Micro Edition), MIDP (Mobile Information Device Profile), CLDC (Connected Limited Device Configuration) palaikantys moduliai. Modulių kombinacija yra viena populiariausių šių laiko mobiliuosiuose įrenginiuose, esant jai į įrenginį galima įdiegti daug naudingų vartotojams programų.

Tikslas – studijuoti optimalumo principais pagristus įvairius modelius, jų analizės metodus, algoritmus įvairiose srityse kylantiems optimizavimo uždaviniams spręsti, pritaikant juos mobiliojo telefono aparato aplinkai. Praktinė užduotis – naudojant MIDP v2.0 sukurti *Pareto* optimumo metodu paremtą plano pasirinkimo programą, mobiliajam telefonui. Darbo metu siekiama įsisavinti MIDP v2.0 ir CLDC v1.1, gilinti optimizacijos teorijos žinias, išmokti taikyti matematikos ir informatikos metodus sprendžiant uždavinius, studijuoti optimizacijos uždavinių klasės bei jų sprendimo metodus, įgyti sugebėjimus, vertinant praktinius uždavinius, jų sudėtingumą, bei pasirinkti tinkamiausius metodus jiems spręsti.

Trečios kartos mobilieji įrenginiai turės daug daugiau galimybių, tai lems didėjantį jų vartojimą bei paslaugų tokiems įrenginiams kūrimą, todėl reikės skirti nemažai dėmesio duomenų apsaugai. Sun Microsystems pristatė platformą tokiems įrenginiams – Java 2 Micro Editon. J2ME teikia lanksčias taikomąsias programas vartotojams ir įtvirtintoms sistemoms, kuri jungia ir mobiliuosius įrenginius. Vartotojas turintis įrenginį su J2ME, gali parsisiųsti programas iš atviro tinklo ir naudotis jomis savo mobiliajame įrenginyje.

2. Optimizacinių metodų analizė

2.1. Optimizavimo sąvoka

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

Dažnai nagrinėjama aibė yra *n*-matės Euklido erdvės poaibis. 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 tenkinti 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 nurodyta, kad kintamasis x_i kinta $0 \le x_i \le 5$, tai akivaizdžiai matysime, kad apatinis rėžis 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.:

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

Minimizavimo uždavinį galima būtų užrašyti taip:

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

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 labai pasikeisti, pasikeitus duomenims visai nežymiai.

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

Sprendžiamas klausimas: Kaip taikyti ir naudotis optimizavimo teorija sprendžiant optimizacijos uždavinius? Norint taisyklingai suformuluoti uždavinį, reikia sudaryti taikomojo uždavinio modelį:

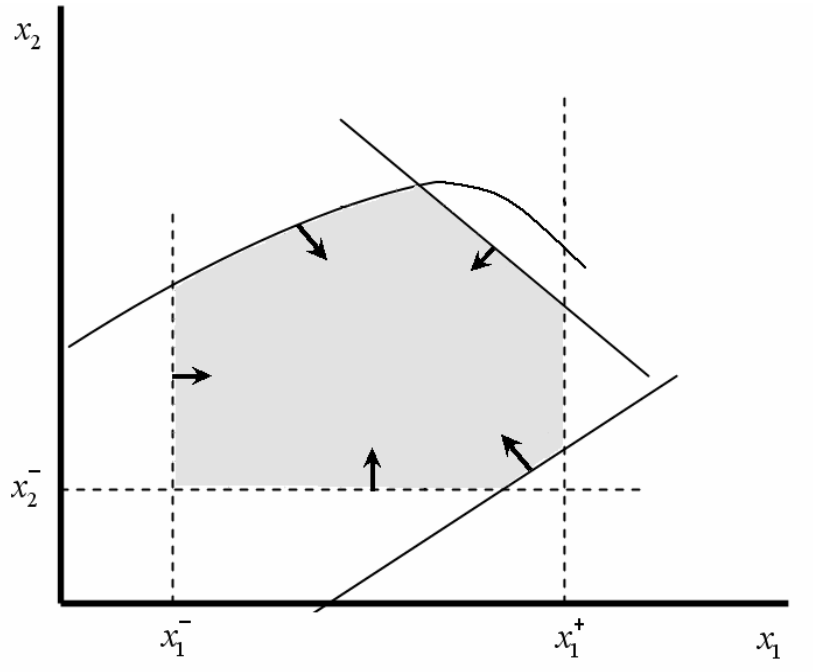
- Išsiaiškinam, kokį kriterijų reikėtų optimizuoti, taip pat nustatome ribojimus, t.y. kokie uždavinio parametrai gali būti keičiami ir, kokioje srityje jie gali kisti. Reikia apibrėžti kintamųjų vektorių *X*, tikslo funkcija $f(X)$ ir leistiną sritį *A*. Kai panaudosime pasirinktą optimizavimo metodą,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

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

Naudodamiesi optimizacijos modelio rezultatais ir aprašymais galėsime parinkti optimalų metodą optimizacijos uždavinio sprendimui.

2.2. Pareto optimalumas

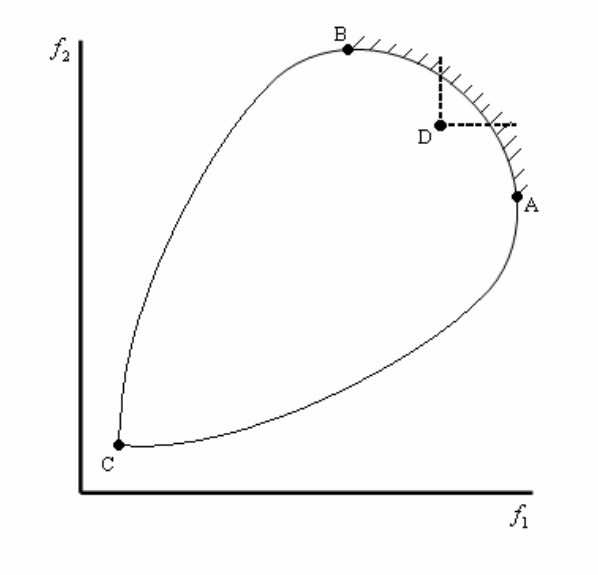
Tarkime turime vektorių $X = (x_1,...,x_n)$, kuris turi *n* keičiamų parametrų. Jis parenkamas leistinoje srityje (arba aibėje) $A \in R^n$. Kai kuriems kintamiesiems mes galime apibrėžti apatinę, kai kuriems viršutinę, o kai kuriems, ir abi ribas. Pavyzdžiui, leistina sritis (*2 pav.*) apibrėžta trimis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



1 pav. Leistinosios aibės pavyzdys

Kiekvienam vektoriui *X,* apibrėžiamas kriterijų vektorius $(f_1,...,f_m) = F(X)$. Vektoriui *X* kintant aibėje *A* vektorius *F* perbėga į 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 minimumo taškai paprastai nesutampa.

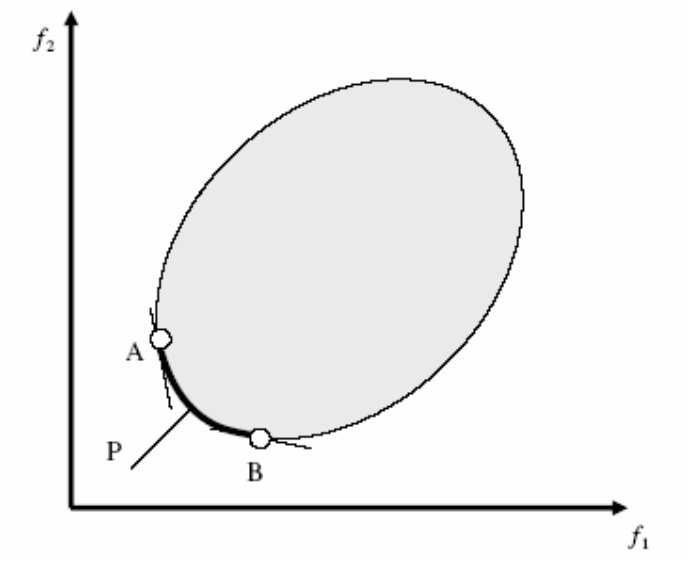
Pavaizduokime dviejų kriterijų reikšmių aibę (*3 pav.*), pirmojo kriterijaus maksimumo tašką *A* ir antrojo kriterijaus maksimumo tašką *B*, kurie nesutampa. Tačiau sutampa abiejų kriterijų minimumai
taške *C*.



2 pav. Kriterijų reikšmių aibės pavyzdys

Daugelio kriterijų uždavinių teorijoje paprastai ieškoma visų kriterijų optimalumo. Vieno kriterijaus optimizavimo uždaviniuose tikslo funkciją minimizavome. Pateiktas dviejų kriterijų maksimizavimo uždavinys *(2 pav.*) gali atrodyti prieštaringas. Jei, kaip sprendinį imsime tašką *A,* tai gerokai nukrypsime nuo maksimalios antrojo kriterijaus reikšmės, ir atvirkščiai. Todėl reikia ieškoti kompromisinio varianto. Antra 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 (*2 pav.*) užbrūkšniuota.

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. Kiekvienas užbrūkšniuotas (*3 pav.*) kreivės taškas nepagerinamas. Išanalizuokime dar vieną situaciją pavaizduotą (*3 pav.*):



3 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 gerinti ir jo negalime vadinti optimaliu. Jeigu paimsime tašką, priklausantį kreivei *P*, pamatysime, kad nejudinant taško didės vienas iš kriterijų. Čia galima pamatyti, kad egzistuoja daugybė Pareto taškų, kurie kreivėje *P* yra pasiskirstę tarp taško *A* ir *B* .

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

$$f_i(X) \le f_i(X^*), i = 1,...,m \qquad (1.1)$$

ir bent vienam *j,*

$$f_j(X) < f_j(X^*). \qquad (1.2)$$

Efektyvaus sprendinio kriterijų vektorius nepagerinamas, ir atvirkščiai: kiekvieną nepagerinamą kriterijų (Pareto) vektorių atitinka efektyvus sprendinys. Pavyzdžiui, jei pateikiami trys galimi mobiliųjų planų variantai $X_i, i = 1,...,3$,

besiskiriantys kriterijų vektoriaus $F(X) = (f_1(X),..., f_5(X))$ reikšmėmis, kurių reikšmės:

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

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

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

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

2.3. Vektorinės optimizacija

Racionalus išrinkimas gali būti pakeistas naudingumo funkcijos maksimizavimu (arba minimizavimu). Tačiau ir nežinant šio rezultato atrodo, kad gan natūralu pakeisti daugelio kriterijų uždavinį viena kriteriniu, sudarytu susumavus viso uždavinio kriterijų svorius

$$f(X) = \sum_{i=1}^{m} a_i f_i(X), \sum_{i=1}^{m} a_i = 1, a_i \geq 0. \qquad (1.3)$$

Pavyzdžiui, jei paimsime tą pati pavyzdį su mobiliaisiais planais, kuris paminėtas anksčiau, tai matysime, kad renkant planus pagal optimalų Pareto kriterijų, bus atmesti visi neracionalūs planai: paskui iš optimalių Pareto planų galima bus išsirinkti tinkamą. Kaip tokį uždavinį galima spręsti pritaikius svorius?

Jei paimsime realią situaciją, tai matysime, kad ne visada žmogui yra svarbūs tam tikri parametrai. Pavyzdžiui, perkant mašiną, vienam svarbu, kad būtų galingesnis variklis, kitam, kad mašina atrodytų gražiai ar pan. Todėl grįžtant prie mobiliųjų planų uždavinio galima padaryti išvadas, kad jeigu vienam svarbiau mažesnė kaina skambinant, tai kitam gali rūpėti visai kiti kriterijai, pvz. jam svarbiau, kokios žinutės kaina į kitus tinklus. Tuomet programoje, kuri išrenka optimalų variantą, pirmasis galėtų nurodyti didesnį svorį skambučio kainai, o antras – žinutės. Jeigu, sprendžiant pagal Pareto optimalumą (1.1) (1.2), bus atmesti tik neracionalus planai, tai naudojant

vektorinę optimizaciją su svoriais, galima pagal individualias užgaidas pritaikyti mobilųjį planą.

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

Paprastai, psichologiniu eksperimentu, pavyksta nustatyti tik nedideliam alternatyvų skaičiui. Todėl galima tikėtis parinkti svorius (1.3) taip, kad funkcijos *f(X)* reikšmių santykiai atitiktų eksperimentinius duomenis. Antra vertus, kartais kriterijų svorius galima nustatyti, atsižvelgiant į kriterijų svarbą.

Panagrinėsime svertinės kriterijų sumos savybes:

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

$$\max_{X \in A} \sum_{i=1}^{m} a_i^0 f_i(X) = \sum_{i=1}^{m} a_i^0 f_i(X^0) \qquad (1.4)$$

Šis S. Karlino rezultatas teoriškai pagrindžia svertinė svorių kompozicijos taikymą.

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

$$\sum_{i=1}^{m} a_i f_i(X^*) = \max_{X \in A} \sum_{i=1}^{m} a_i f_i(X) \qquad (1.5)$$

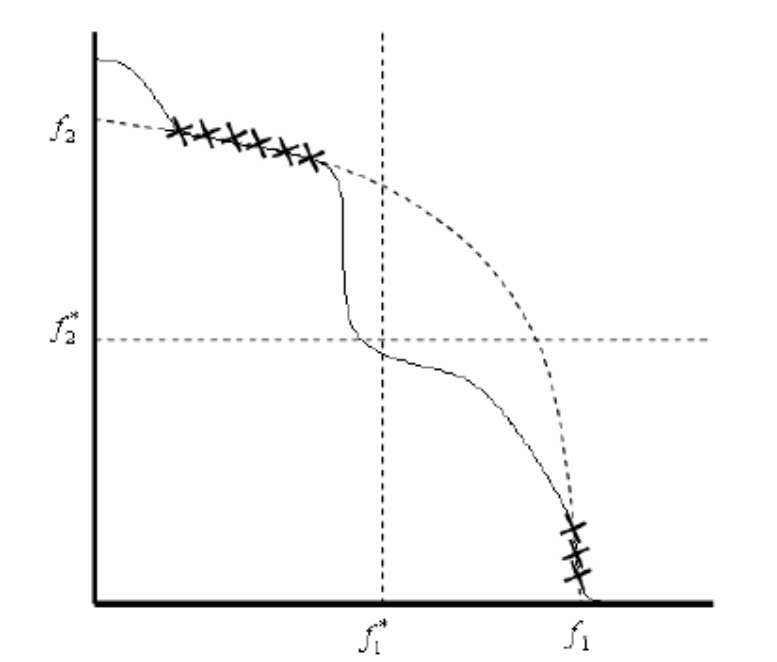
tai X^* – efektyvus sprendinys.

tai X^* yra efektyvus sprendinys.

Iš šių teoremų matome, kad maksimizuojant svertinę kriterijų sumą, visada gaunamas efektyvusis sprendinys. Atitinkamai parinkdami svorius, galime gauti bet kokį efektyvųjį sprendinį. Gali susidaryti įspūdis, kad naudodamiesi šia kriterijų kompozicija, galėsime išspręsti visas daugelio kriterijų optimizavimo problemas. Tačiau iškilumo reikalavimas Karlin'o teoremoje susiaurina jos pritaikymo sritį, nes kriterijų iškilumas projektavimo uždaviniuose gali būti abejotinas ir tik tam tikrais atvejais įrodomas. Jei iškilumo sąlyga yra priimtina, tada kyla problema, 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 atitinkantys kriterijų vektoriai gali būti pasiskirstę Pareto aibėje labai netolygiai. 3 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 $f_i > f_i^*$. 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 daugelio kriterijų 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.



4 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 nepriimtinos kriterijaus reikšmės gali ir nebūti lygios nuliui. Tačiau, pakeitus skalės atskaitos 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:

$$f(X) = f_1(X) \cdot f_2(X) \cdot \ldots \cdot f_m(X). \qquad (1.6)$$

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

$$f(X) = \min_i a_i \cdot f_i(X) \qquad (1.7)$$

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

Atlikus tyrimus nustatyta, 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 rinkdamasis prekę pirmiausia pažiūrėti kokia tos prekės kaina, o tik paskui atsižvelgti į kitus prekės parametrus ir jeigu kitas parametras jam nepatiks pirkėjas iš karto gali tą prekę atmesti.Tinkamiausi variantai išrenkami nuosekliai atmetant netinkamus, atskiru 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 leksikografinį 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 $A_i \subset A$:

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-tasis 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.

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 leidžiamos 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. Analizės išvados

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 taškas, tik kai kuriais atvejais optimizavimo uždavinį pasiseka išspręsti analitiškai, t.y. formule užrašyti sprendinį. Minimumo apibrėžimas bendru atveju nelabai kuo gali padėti, nes tęstinos (t.y. aibės visu reikšmių negalima patikrinti

netgi atlikus be galo daug veiksmų) leistinosios srities atveju neįmanoma palyginti nagrinėjamos tikslo funkcijos reikšmės su visomis kitomis. Kartais potencialiu sprendiniu aibė yra baigtinė, nors leistinoji sritis ir tęstina. Taip yra, pavyzdžiui, tiesinio programavimo uždaviniuose, kuriuose leistinoji aibė yra tęstina. Surasti optimalu sprendinį baigtinėje potencialių sprendinių aibėje galima atlikus pilna aibės peržiūra, t.y. baigtiniu žingsniu skaičiumi, nors pastarasis gali būti ir labai didelis.

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

3. Optimizacinių uždavinių sprendimo modelis

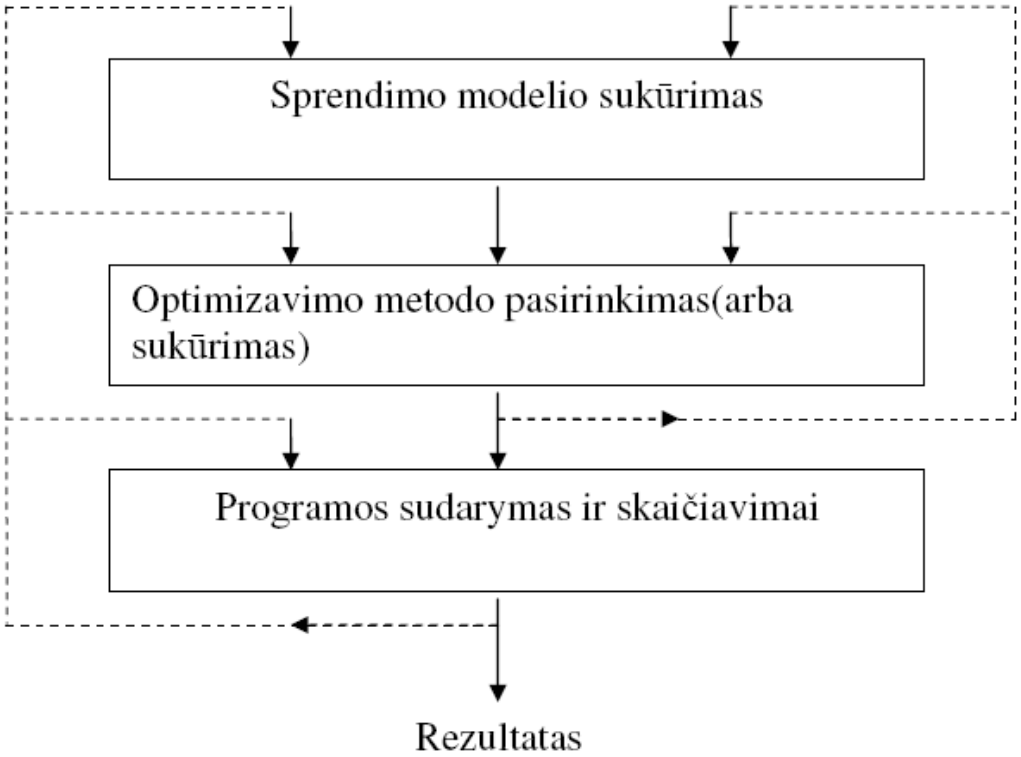
3.1. Formalus optimizavimo uždavinio aprašas

Dažnai realus optimizavimo uždavinius yra sunku suformuluoti ir išspręsti. Problemos iškyla ne tik sudarant uždavinio sprendimui reikalingą modelį, bet ir ieškant optimalių parametrų reikšmių. Informatikos, vadybos, ekonomikos bei kitų sričių specialistai, sprendžiantys šiuolaikines problemas, gana dažnai susiduria su sudėtingais optimizavimo uždaviniais, kurių sprendimas reikalauja kiek gilesnio, negu vieno metodo idėja, optimizavimo teorijos suvokimo. Todėl kiekvienam uždavinio sprendimui reikalinga gerai suprasti, uždavinio sprendimui parinkto metodo teoriją ir jo efektyvumą.

Optimizavimo teorija ir optimizavimo metodų bei juos realizuojančios programinės įrangos kūrimas yra aktyviai vystoma mokslo technikos šaka. Programinės įrangos rinkoje egzistuoja daugybė komerciškai platinamų bei nemokamų programinių produktų skirtų programinės įrangos kūrimui. Todėl pasirenkant kūrimo priemones, optimizavimo uždavinio sprendimui, labai naudinga atsižvelgti į jos pritaikymo universalumą.

3.2. Optimizavimo uždavinio sprendimo modelis

Uždavinio formulavimas – svarbus etapas, nuo jo rezultato priklauso sprendžiamumo uždavinio gautų rezultatų patikimumas ir racionalumas. Sprendžiant optimizavimo uždavinį kompiuteriu labai patogu būtų sudaryti uždavinio modelį, kad patogiau pereitume prie tolimesnių žingsnių. Paveiksle (*1pav.*) matome kaip schemos pagalba pavaizduoti uždavinio kūrimo etapus.



4 pav. Optimizavimo uždavinio sprendimo modelis

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.

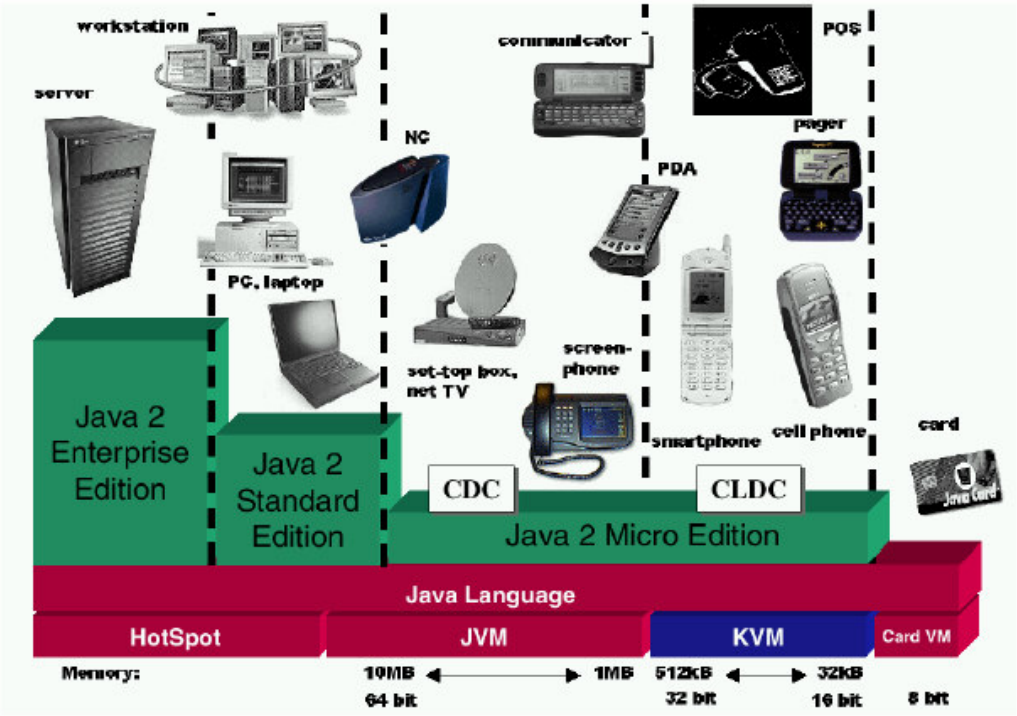
Tobulėjant technologijoms bei esant šiuolaikiniam mokslo lygiui efektyvų sprendimo metodą sunku pritaikyti kiekvienam optimizacijos uždaviniui. Todėl formuluojant uždavinį reikia atsižvelgti į naudojamų optimizavimo metodų efektyvumą.

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

4. J2ME priemonių tyrimas

4.1. J2ME priemonės mobiliuosiuose įrenginiuose

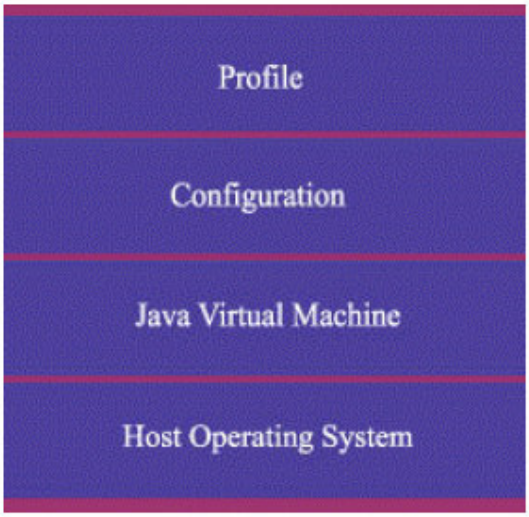
Bendrovė "Sun Microsystems", 1999m. vykusioje kasmetinėje “JavaOne” konferencijoje pristatė mobiliems įrenginiams pritaikytą "Java" versija - "Java 2 Micro Edition" (J2ME). Iš esmės – tai "apkarpytas" standartinis "Java" platformos ("Java 2 Standard Edition") variantas, sukurtas atsižvelgiant į visus mobiliųjų prietaisų apribojimus (mažą atmintį, ribotas procesoriaus galimybes, grafinės sąsajos, informacijos įvedimo/išvedimo ypatumus, nepastovų ryšį su 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 padės įgyvendinti viską, kas tik programuotojui ateina i 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, automobilinėse telemetrijos sistemose ir kišeniniuose kompiuteriuose, užtikrina nepaprastą šių aparatų lankstumą ir padeda vartotojui efektyviau 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" sukurta "Java" programa, leidžiančia 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.



5 pav. Java platformos

Šiandien turint su J2ME suderinamą mobilųjį telefoną, galima naudotis internete populiaraus pranešimų gaviklio ICQ funkcijomis (jeigu operatorius teikia tokią paslauga), su "Bluetooth" pagalba valdyti "Winamp" grotuvo funkcijas, telefoną paversti į judesius reaguojančią apsaugos kamerą (jei jame įmontuota skaitmeninė kamera) ir taip identifikuoti save GSM tinkluose. Pastaroji savybė ypatingai 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 bazė, 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.

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 ypatingai vertinamas. Bent šiek tiek susidūrusieji su "Java" kalba gerai žino, kad savo sintakse ji labai primena C++ programavimo kalbą. Tačiau esminis 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 ta pačia "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.



6 pav. J2ME architektūra

4.2. MIDP ir CLDC

Nustatymai apibrėžia bazinę J2ME aplinką. Ši aplinka apima virtualią mašina, kuri gali būti labiau ribojama negu naudojama J2SE, ir klasių rinkinį įgytą tiesiai iš J2SE. Esmė tame, jog visi nustatymai sukabinami specifinėmis įrenginių šeimomis su panašiomis galimybėmis.

Apibrėžiami tokie nustatymai: *Connected Device Configuration* (CDC) ir *Connected Limited Device Configuration* (CLDC). CLDC Nukreiptas į mažuosius įrenginius: mobiliuosius telefonus, delninukus (PDA), interaktyvius pranešimų gaviklius. Kaip grupė šie įrenginiai turi srovės, atminties ir tinklo pralaidumo suvaržymų, tai tiesiogiai įtakoja kokias Java programas šie įrenginiai gali naudoti.

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 i mobilųjį įrenginį) galimybes.

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 laisvą atmintį iki 512 Kb. Egzistuoja dvi CLDC versijos 1.0 ir 1.1. CLDC 1.1 turi daugiau galimybių, pavyzdžiui – slankaus kablelio palaikymas.

Daugelyje rinkoje esančių mobiliųjų įrenginių 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 telefonu 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 įrenginiuose, nuo standartinių jungčių iki naujosios "Pop-Port" jungties, užtikrinančios ne tik spartų duomenų perdavimą, bet ir sugebančios identifikuoti skaitmeninius priedus bei perduoti erdvinį garsą.

Patobulinta mobiliųjų prietaisų specifikacija "MIDP 2.0" pasirodė 2003 metais ir todėl šiandien dauguma mobiliųjų telefonų gamintojų jau priėmė šį įvairialypį standartą. Galimybės darbui su garsu, grafika, padidintas saugumas, patogumas. MIDP užduodamos mobiliųjų įrenginių charakteristikos:

- ekrano skiriamoji geba nuo 96x54 pikselių (ekrano taško gylis 1 bitas);
- įvedimo įrenginiu gali būti klaviatūra arba lietimui jautrus ekranas;

- 32 Kb dinaminės atminties;
- 128 Kb MIDP komponentams;
- 8 Kb duomenims saugoti;
- bevielis tinklas;
- mažas energijos šaltinis.

4.2.1. CLD Nustatymai

Nustaymai apibrėžia bazinę J2ME aplinką. Ši aplinka apima virtualią mašina, kuri gali būti labiau ribojama negu naudojama J2SE, ir klasių rinkinį įgytą tiesiai iš J2SE. Esmė tame, jog visi nustatymai sukabinami specifinėmis įrenginių šeimomis su panašiomis galimybėmis.

Apibrėžiami tokie nustatymai: *Connected Device Configuration* (CDC) ir *Connected Limited Device Configuration* (CLDC). CLDC Nukreiptas į mažuosius įrenginius: mobiliuosisu telefonus, delninukus (PDA), interaktyvius pranešimų gaviklius. Kaip grupė šie įrenginiai turi srovės, atminties ir tinklo pralaidumo suvaržymų, tai tiesigiai įtakoja kokias Java programas šie įrenginiai gali naudoti.

4.2.2. CLDC Reikalavimai

CLDC nereikia daug resursų. Gali veikti įrenginiuose su 128K ar daugiau netriniąja atmintimi ir 32K ir daugiau triniosios atminties. CLDC įrenginiai turi turėti kokią nors tinklo jungtį, nors tai gali būti su pertrūkiais, ir mažo greičio jungtis. Šis konfigūracija yra apribotiems įrenginiams.

CLDC apibrėžia kiekį reikalavimų Java aplinkai. Pirmasis reikalavimas yra pilnam palaikymui Java kalbos, neskaitant kelių skirtumų. Šie skirtumai:

- **Nėra slankaus kablelio palaikymo.** Slankaus kablelio tipai ir konstantos nėra palaikomos, taip pat nėra palaikomos J2SE branduolio klasės dirbančios su slankaus kablelio reikšmėmis. Metodai paimantys ar gražinantys slankaus kablelio reikšmes yra pašalinti iš visų klasių.
- **Nėra objektų užbaigimo.** Supaprastinant šiukšlių surinkimo užduotį, `finalize` metodas yra pašalintas iš `java.lang.Object`. Šiukšlių

surinkėjas paprastai panaudos bet kurį nenurodytą objektą. Šis veiksmas apsaugo nenukreiptus objektus nuo atgaivinimo jų pačių, taip pat iššaukia papildomą užsakymų darbą šiukšlių surinkėjui.

- **Vykdymo klaidos traktuojamos įgyvendinimo-priklausomumo formoje.** Vykdymo klaidos yra išimtys subklasės `java.lang.Error` mėtomos virtualios mašinos. CLDC tiktai apibrėžia tris šių klaidų klases:`java.lang.Error`, `java.lang.OutOfMemoryError`, and `java.lang.VirtualMachineError`.

Antrasis reikalavimas pilnam virtualios mašinos palaikymui, neskaitant šių skirtumų:

- **Nėra slankaus taško palaikymo.** CLDC įrenginiai gali neturėti giminingo palaikymo slankaus taško operacijoms. Kadangi VM nepalaiko slankaus taško konstantų operacijų arba bet kokių bitų kodų apimančių slankaus taško tipus.
- **Nėra užbaigimo ir silpnos adresacijos.** Tai palikti išorėje, supaprastinti šiukšlių surinkimo algoritmą.
- **Neėra JNI palaikymo ar atspindžio ar bet kokio žemo lygio sąsajos priklausančių nuo jų.** CLDC neturi palaikymo objektų išdėstymui. VM gali turėti įgimtą sąsają, suderinamumo sąsają, profiliavimo sąsają, bet tai nėra reikalujama, ir tai neturi būti standartinė J2SE sąsaja.
- **Nėra gijų grupių ar dameon gijų.** Gijos palaikomos, tačiau gijų grupės ir dameon gijos – ne. VM gali pasirinkti realizuoti gijas priklausomai nuo OS arba atliekant savitą perjungimą.
- **Įgyvendinimu apibrėžtas klaidų apdorojimas.** Bet kurios CLDC neapibrėžtos vykdymo klaidos yra apdorojamos specifiniu metodu.
- **Klasių verifikavimas atliekamas skirtingai.** Standartinis klasių verifikavimo procesas yra per daug brangus, taigi alternatyvus procesas buvo apibrėžtas. Alternatyvus procesas perkelio didesnę verifikavimo dalį atskiram preverifikavimo žingsniui, kuris atsiranda darbalaukio arba serverio kompiuteryje ir ne įrenginyje. Preverifikuotos klasės laikmenos yra paleidžiamos įrenginyje naudojant antrą žymiai paprastesnį verifikavimą, kuris paprasčiausiai validuoja rezultatus preverifikavimo žingsnyje.

Trečiasis reikalavimas – bet kurios klasės įtrauktos ar paveldėtos iš J2SE, privalo būti poaibiai J2SE 13 klasių. Metodai gali būti neįtraukti, bet naujų viešų metodų ar duomenų narių negalima pridėti. Tinkamumas pirmaeilės svarbos.

Ketvirtas reikalavimas – klasės apibrėžtos CLDC ir jo profilio esti `javax.microedition` pakete arba subpakete, kuris leidžia identifikuoti klases specifines CLDC.

Galutinis reikalavimas yra minimaliam suteikimui lygių teisių palaikymui. CLDC teikia bazinį palaikymą konvertuojant bitų srautus į Unicode ir atgal naudojant bent vieną simbolių koduotę. CLDC neperadresuoja lokalizacijos problemų, tokių kaip atvaizduoti datas, laikus, valiutas ir kitas vietovėms būdingas elgsenas.

4.2.3. Palaikomos J2SE klasės

CLDC apima klases ir sąsajas įtrauktas iš šių trijų J2SE paketų:

- `java.lang`
- `java.io`
- `java.util`

Kaip matome dauguma J2SE klasių yra neįtrauktos, įskaitant tokias iš reikalingų paketų tokių kaip `jawa.awt`, `java.net` ir `java.sql`. Netgi paketai kurie esti įtraukti neturi klasių, daugelis tokių klasių yra metodai. Sąrašas palaikomų klasių yra pateiktas lentelėje 1.

PACKAGE	CLASS
java.lang	Boolean
	Byte
	Character
	Class
	Integer
	Long
	Math
	Object
	Runnable
	Runtime
	String
	StringBuffer
	System
	Thread
	Throwable
java.io	ByteArrayInputStream
	ByteArrayOutputStream
	DataInput
	DataInputStream
	DataOutput
	InputStreamReader
	OutputStream
	OutputStreamWriter
	PrintStream
	Reader
	Writer
java.util	Calendar
	Date
	Enumeration
	Hashtable
	Random
	Stack
	Time
	Vector

Lentelė 1: Klasių sąrašas iš J2SE 1.3 įtrauktų į CLDC

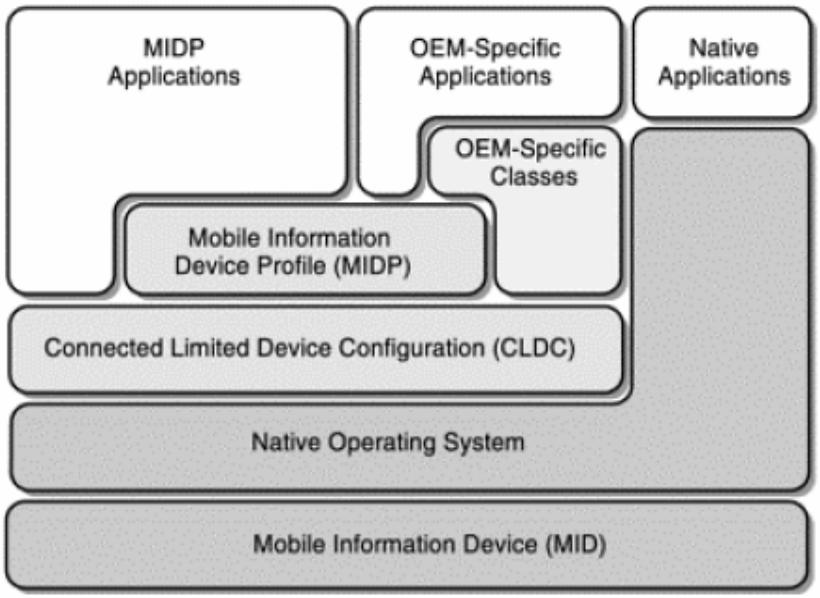
4.2.4. MID profilis

Profilis praplečia nustatymus, pridėdamas specifinių klasių branduoliniam klasių rinkiniui. Kitaip sakant , profiliai tiekia klases kurios reikalingos specifiniam įrenginio naudojimui ir tiekia funkcionalumą, kurio trūksta bazinėje konfigūracijoje – dalykai, tokie kaip vartotojo sąsajos klasės, persidengimo mechanizma ir t.t. Kol jie teikia svarbų ir reikalingą funkcionalumą, nekiekvienas įrenginys palaikys kiekvieną profilį. Pavyzdžiui Japonijoje NTT DoCoMo išleido kiekį įrenginių palaikančių Java Su CLDC bet su savo nuosavu profiliu. Programos parašytos šiems įrenginiams nedirbs mobiliuosiuose telefonuose palaikančiuose MIDP.

Yra jau sukurtų profilių ir yra vystomų. Be MIDP, kuris yra paremtas CLDC, sekantys profiliai yra arba bus prieinami:

- *Personal Digital Assistant Profile* kuris praplečia CLDC perimti pranašumą praplėsto galimybėmis PDA lyginant su paprastesniais įrenginiais su MIDP.
- *Foundation Profile* Prideda papildomas klases prie CDC, bet ne vartotojo sąsajos klases. Pasitarnauja kaip pagrindas kuriant naujus profilius.
- *Personal Profile* kitaip apibrėžia asmeninės Java kaip J2ME profilį. Taip pat praplečia Foundation Profile.
- *RMI Profile* prideda RMI palaikymą prie CDC.

Sudėtiniai profiliai gali egzistuoti su tais pačiais nustatymais. Profiliai taip pat gali būti kuriami vienas ant kito – pvz. Personal Profile yra praplėtimas Foundation Profile.



7 pav. MIDP architektūra

4.3. K virtuali mašina

KVM buvo kuriama, kad būtų kuo galimai mažesnė ir efektyvesnė, tvirta Java kalbos branduolyje. Atmintis tokiuose įrenginiuose labai brangus dalykus, kadangi jos apytiksliai tėra 60 kilobitų. Tai paaiškina, kodėl *KVM* yra raidė *K*, virtuali mašina veiks tik su keliomis dešimtimis kilobitų atminties. Kaip mini *Sun, KVM* buvo parašyta C kalba, siekiant išgauti lankstumą. Dauguma kodo sudarančio *KVM* nesikeis naudojant kitą OS. Įrenginių priklausomybė kodui, lemia atminties valdymas, susitvarkymas su kritinėmis klaidomis, įvykių seka, kodo inicializavimu ir išsivalymu.

4.4. Saugumas

Galimybė savarankiškai kurti programas ir laisvai diegti bei paleisti jas "Java" telefonuose sąlygoja neapdairiai paliktas saugumo spragas, kurios išradingų įsilaužėlių ir virusų kūrėjų rankose labai greitai virsta klastingais spąstais telefono savininkui. 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 įdiegti keli "saugikliai". Visų pirma "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 ypatingai 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 naudojamu (tik telefonams suprantamu) kodu pagalba pavyko "pakabinti" telefoną, tačiau dėl to buvo kalta ne "Java", o telefone naudojama gamintojo mikroprograma. 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 žkrato 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.

4.5. J2ME alternatyva – BREW platforma

Kai belaidė "Java" jau aktyviai skynėsi kelia į mobiliųjų įrenginių rinką, pirmuosius žingsnius žengė ir tuo metu grėsmingiausias jos varžovas – korporacijos "Qualcomm" sukurta BREW (Binary Runtime Environment for Wireless™) platforma. Tai jungiamoji grandis tarp mobiliojo įrenginio procesoriaus ir C/C++ kalba parašytų programų. Kaip ir "Java", BREW yra atviras standartas, suteikiantis programuotojams visišką kūrybinę laisvę. Pagrindiniai jos privalumai – mažesni kuriamų programų reikalavimai įrenginyje naudojamos atminties dydžiui, galimybė kurti sudėtingas programas darbui su dideliais duomenų masyvais (naudojantis 3G mobiliojo ryšio tinklais) 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 besinaudojanciųjų 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. Šiai dienai kai įvyko techninės telefonų įrangos standartizacija "Java" turi didelį pliusą savo rankose. Sparčiai išpopuliarėjus daugialypės terpės telefonams turintiems standartinę ekrano skiriamąją gebą, galima visapusiškai išnaudoti visus J2ME privalumus. Todėl kol kas J2ME praktiškai neturi konkurentų. Lieka tik stebėti įvykius ateityje.

4.6. Įrašų valdymo sistema

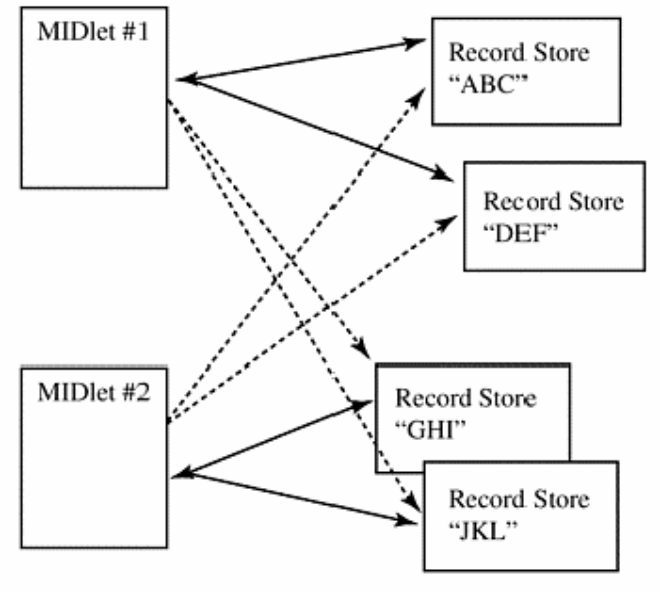
Pradiniame savo darbo variante, buvau sugalvojęs sistemą, kurioje informacija, būtų gaunama ir patalpinama į tekstinę rinkmeną. Tačiau detaliau panagrinėjus ir parašius pirmąjį programos variantą, susidūriau su kliūtimi – duomenis nustatyti iš tekstinės rinkmenos nėra problemų, bet patalpinti rezultatų – programa neleidžia, kadangi to pasakoje pasikeis JAR failo dydis. Tuomet teko ieškoti alternatyvų. Daugelis programavimo kalbų palaiko darbą su DB, J2ME ne išimtis, joje įdiegta – įrašų valdymo sistema (RMS).

RMS – MIDP palaikoma pastovi aplinka. RMS yra įrašams orientuota DB, kurioje egzistuoja eilutės turinčios savo unikalų ID (*8 pav.*).

Record ID	*Data*
1	Array of bytes
2	Array of bytes
3	Array of bytes
...	...

8 pav. Įrašų saugykla

Kiekvienas MIDlet'as gali turėti daugiau negu vieną įrašų saugyklą, tačiau lentelių vardai būtinai privalo būti skirtingi (*9 pav.*)



9 pav. Priėjimas prie įrašų saugyklų

4.7. Analizės išvados

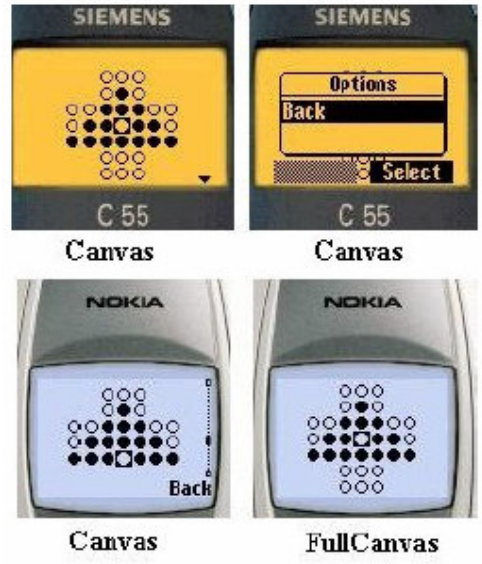
Detaliau panagrinėjus visus J2ME programinės įrangos naudojimo mobiliuosiuose įrenginiuose niuansus, darau prielaidą, kada tai terpė, kurioje:

- lengva diegti ir šalinti programas;
- galimybė kurti nedidelėms, bet funkcionalioms programoms;
- pasižymi universalumu;
- neribotos pritaikymo galimybės;
- didžiulis nepriklausomu programinės įrangos kūrėju susidomėjimas;
- nuolat didėjantis naudingu programų asortimentas;
- beveik visi naujausi telefonai turi įdiegta J2ME technologiją;
- įrenginiai su J2ME nedideli ir patogus transportavimui.

5. Programos Optimalus Pareto planas mobiliosiose aplinkose tyrimas

5.1. Techniniai įrenginio parametrai

Atsiradus dideliam mobiliųjų telefonų gamintojų skaičiui, kiekvienas iš jų mato savo ateitį standartizuojant savo programinę įrangą. Todėl rinkoje egzistuoja daugybė programinės įrangos, skirtos, beveik kiekvienam, egzistuojančiam telefono modeliui. Šioje sferoje pirmauja suomių bendrovė „Nokia“, diktuojanti madas mobiliųjų telefonų rinkoje. Teoriškai kiekvienas programinis produktas, sukurtas J2ME pagalba, turėtų veikti pasirinktame mobiliajame telefone.



10 pav. Skirtumai tarp Siemens ir Nokia

Kiekvienas telefonų gamintojas neišvengiamai naudoja tik sau priimtinas konfigūracijas. Labiausia tai susiję su telefono ekranu, bei grafiniu interfeisu. Pavyzdžiu: „Siemens“ mobilieji telefonai, su spalvotu ekranu, turi skiriamąją gebą 101x80, tuomet kai dauguma „Nokia“ spalvotų telefonu – 128x128 pikselių ir kiekvienas telefonas skirtingai vaizduoja vieną arba kitą grafinį elementą. Tarkime kai naudosime piešimą ekrane, objekto Canvas pagalba, tai Siemens piešimui skirs visa ekraną, komandas paslėpdamas apačioje trikampio pavidalu. Tuo tarpu Nokia neskiria piešimui viso ekrano ir komandas tiesiogiai parodo ekrano apačioje. Nokia yra galimybė naudoti objektą FullCanvas, kuris skiria piešimui visa ekraną, bet tuomet

negalima įdėti jokių komandų. Todėl, kuriant programinį produktą pritaikytą konkrečiam mobiliojo telefono modeliui, reikėtų atsižvelgti į techninius mobiliojo įrenginio parametrus.

5.2. Mobiliųjų įrenginių emuliatoriai

Virtualus telefono emuliatorius – tai programa, imituojanti telefoną ir jo palaikomas funkcijas, ja galima kompiuteryje paleisti programas, sukurtas mobiliajam telefonui, atsižvelgiant į konkretaus telefono modelio techninius parametrus. Daugelis mobiliųjų telefonų gamintojų išleidžia, telefono programavimui skirtas, priemones, kurios vadinamos *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“ padalinys visus telefonų modelius skirsto į platformas pagal serijas (Series Developer Platform): 40, 60, 80 ir 90 serijos. Telefonai į serijas skirstomi atsižvelgiant į konkretaus telefono modelio ekrano skiriamąją gebą:

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

Telefonai priklausantys 40 serijai, 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 priklausantys modeliai, turi Symbian operacinę sistemą ir leidžia programuotojui kurti programas ne tik naudojant J2ME, bet ir C++ programavimo kalbą.

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

Taip pat vertėtų paminėti kitus populiarius mobiliųjų telefonų gamintojus tokius kaip „Motorola“, „Samsung“, „Sony Ericsson“, kurie pateikia įvairius emuliatorius bei pagalbines priemones, padedančias kurti bei eksploatuoti turimus mobilius telefonus.

11 pav. „NOKIA“ telefonų emuliatoriai

5.3. Programos įgyvendinimui naudotos priemonės

Programinė įranga, naudota kuriant programą:

- JavaTM 2 SDK, Standard Edition – Java bibliotekos v1.4.2;
- 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 telefonu emuliatoriai:

- J2ME
- Nokia
- Sony Ericsson
- Motorola

- Siemens

Sisteminiai reikalavimai:

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

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

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

Miminalūs „NetBeans 4.0“ sisteminiai reikalavimai:

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

- Microsoft Windows operacinė sistema:
    - Procesorius: 500 MHz Intel Pentium III
    - Operatyvi atmintis: 384 megabytes
    - Laisvos vietos diske: 125 megabytes

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

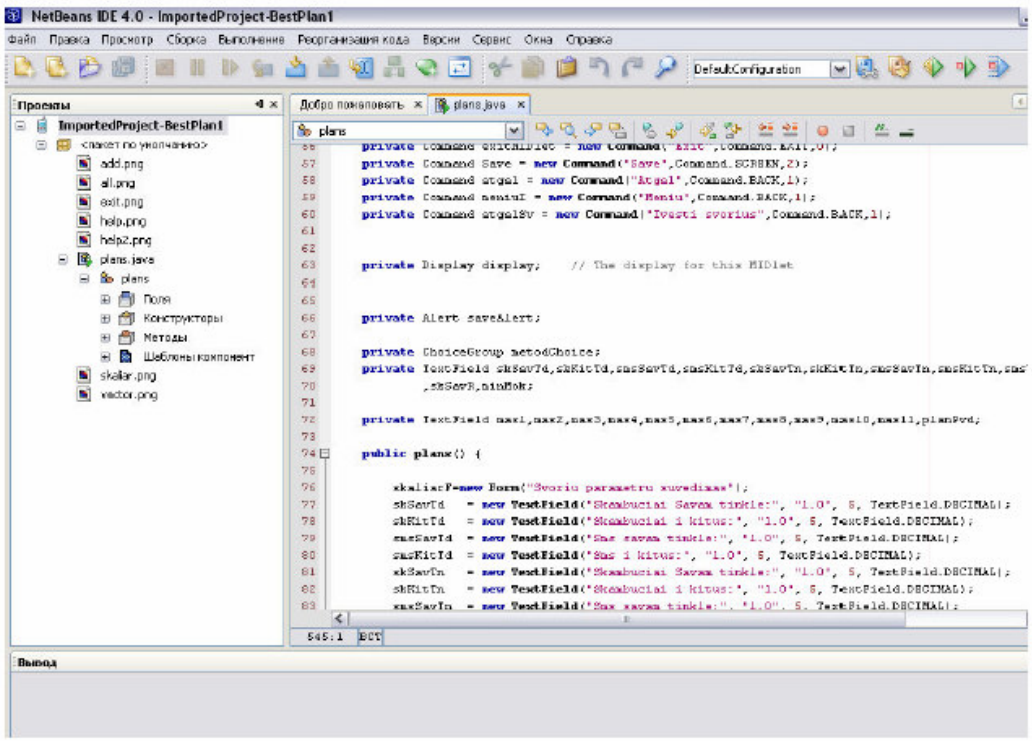
- Macintosh OS X operacinė sistema:
    - Procesorius: PowerPC G4
    - Operatyvi atmintis: 512 megabytes
    - Laisvos vietos diske: 125

Reikalavimai mobiliajam telefonui:

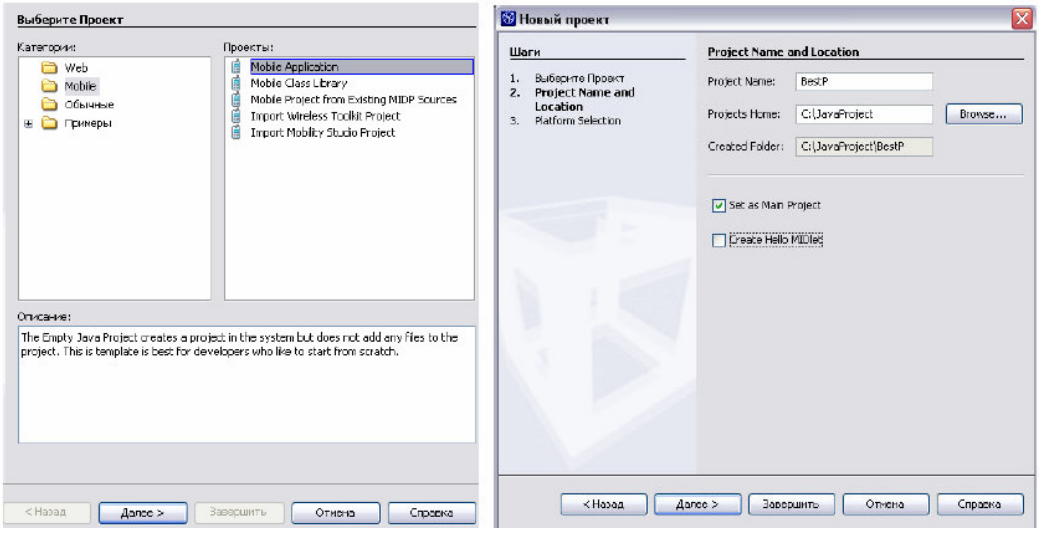
- Java palaikymas
- MIDP 2.0 profilis
- CLDC 1.1 konfigūracija

5.4. „NetBeans“ kūrimo aplinka

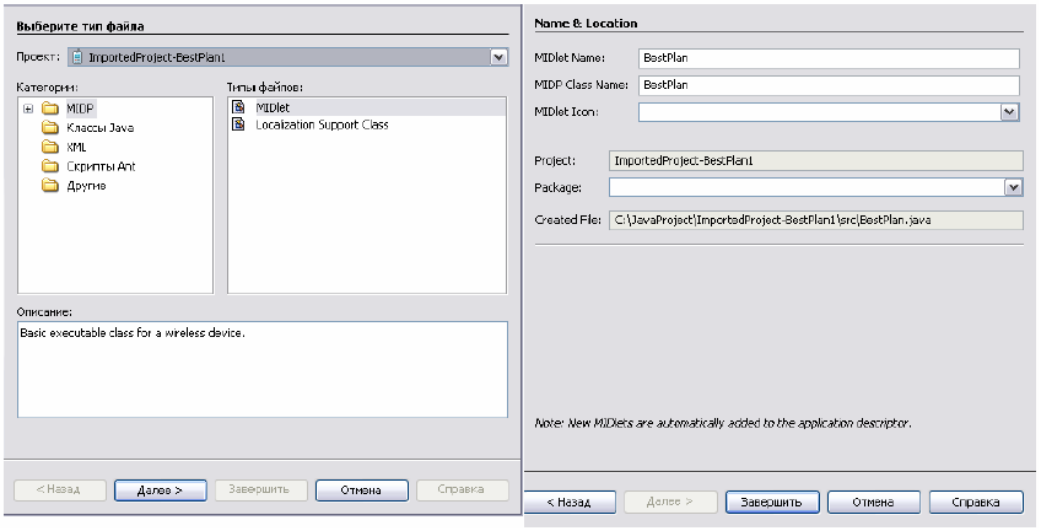
Žemiau pateiksiu, programos „Pareto optimumas“ kūrimui naudotos „NetBeans“ programos aplinką.



12 (a) pav. „NetBeans“ programos aplinka



12 (b) pav. „NetBeans“ programos aplinka. Projekto kūrimas



12 (c) pav. „NetBeans“ programos aplinka. Rinkmenos priskyrimas projektui

Prieš paleidžiant programą, pasirinkus projekto ypatybes, galima jį suderinti su konkretaus modelio mobiliojo telefono emuliatoriumi. Mano atveju buvo naudota standartinė konfigūracija, naudojant J2ME standartinį „DefaultColorPhone“ emuliatorių, kuris yra integruotas į „J2ME Wireless Toolkit 2.3“ paketą.

5.5. Panaudoti J2ME paketai

Kuriant programą buvo panaudoti šie Java 2 ME profilio MIDP 2.0 ir CLDC 1.1

konfigūracijos paketai:

- *import javax.microedition.midlet.*;*
- *import javax.microedition.lcdui.*;*
- *import javax.microedition.rms.*;*
- *import java.io.*;*
- *import java.util.*;*
- *import java.lang.*;*

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

Java.microedition.lcdui – paketas turi savyje įvairias klases, programos sąsajos mobilioje aplinkoje realizavimui.

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

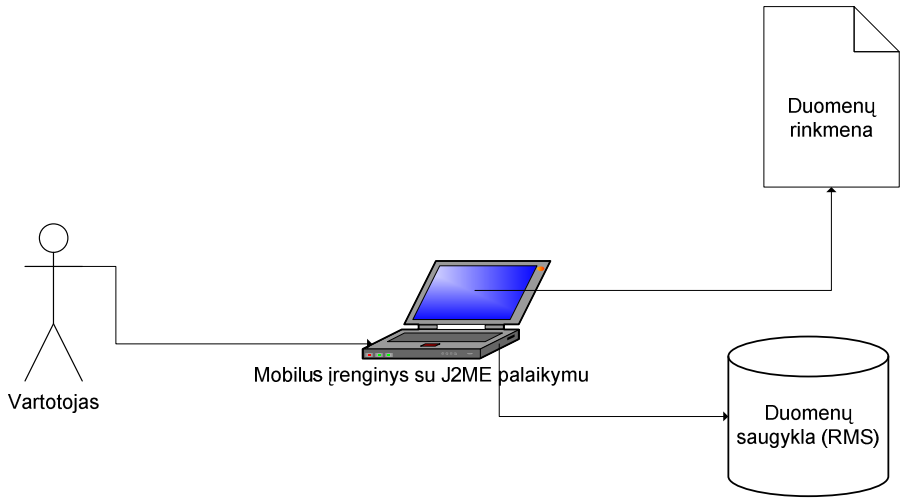
Javax.microedition.rms – paketas skirtas valdyti įrašus DB.

Java.io – paketas skirtas darbui su informacijos duomenų srautais.

Java.util – apima daug naudingų funkcijų, tačiau tai yra vienas iš apkarpytų J2ME paketų, taigi nemažai funkcijų skirtų darbaui su eilutėmis jame nėra.

5.6. Sistemos prototipas

Prototipo architektūra pateikta žemiau esančiame paveiksle:



13 pav. Sistemos prototipas

Pagrindinė sistema kuriama naudojant J2ME ir ją palaikančius paketus.

Duomenys saugomi duomenų bazėje, kurios struktūra atkartoja tekstiniuose failuose kaupiamų mobiliųjų planų informaciją.

DB lentelės struktūra informacijai saugoti

Įrašo ID	Įrašas
1	Tekstinė eilutė, su parametrų pavadinimais
2	Tiekėjas, Planas, Plano tarifai
....	.....
N	.....

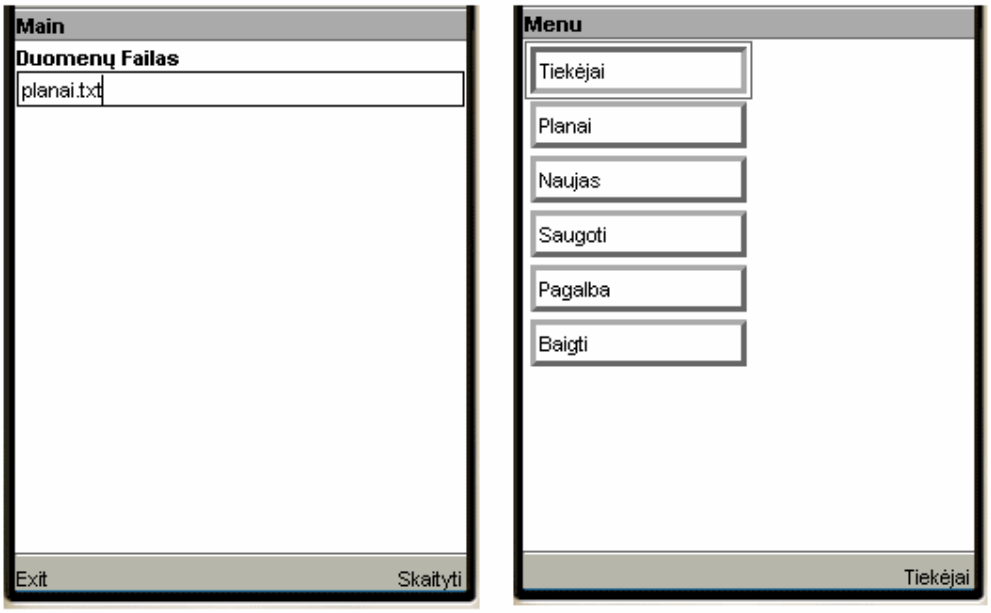
2 lentelė Lentelės struktūra

5.7. Vartotojo dokumentacija

Startavus J2ME emuliatorių WTK 2.3 ekrane pasirodo pagrindinis programos langas.

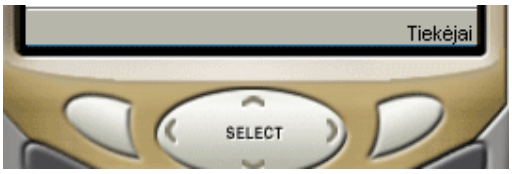
14 pav. Pagrindinis programos langas

Paleidus programą matome pasirenkamą aplikaciją „Pareth“. Ją paleisti spaudžiame mygtuką „Launch“ (*14 pav.*). Paleidus „Pareth“ telefono ekrane pamatysime pasirinkimo „Meniu“(*15 pav.*):



15 pav. Langas pasirodo startavus programai

Jei programa jūsų įrenginyje įdiegta pirmą, kartą, tuomet paleidus programą ekrane pirmiausiai pasirodys langas, prašantis užkrauti duomenis iš duomenų rinkmenos į duomenų bazę. Programai paruošiau di duomenų rinkmenas su naujais duomenimis („*planai.txt*“) ir senais („planai2.txt“). Programai nuskaičius duomenis iš rinkmenos, būsite nukreiptas į programos menu, taip atsitiks ir tuo atveju jei jūsų DB nebuvo tuščia.



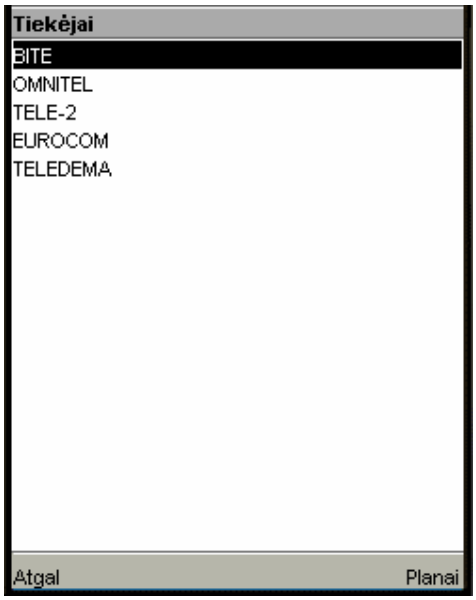
16 pav. Navigacijos bei patvirtinimo klavišai

Menu:

- **Tiekėjai**

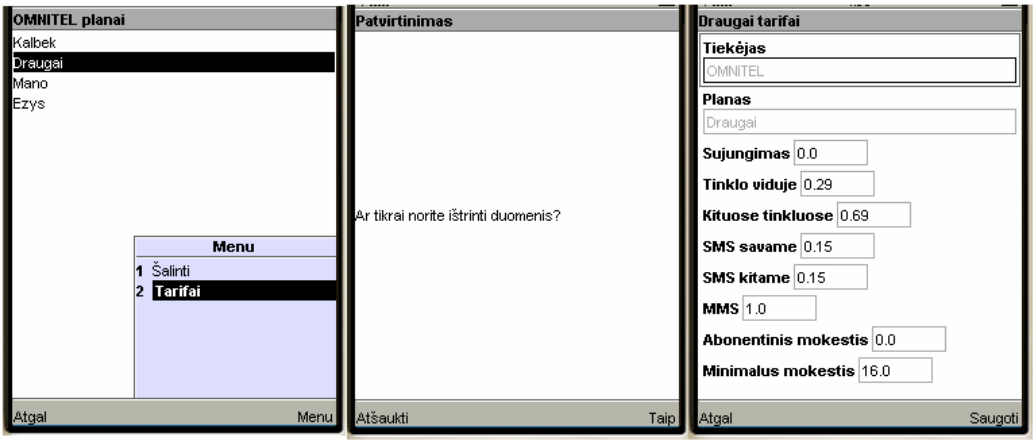
  Pasirinkus šį menu punktą, programa, atvaizduos visus turimus mobiliojo ryšio operatorius (*17 pav.*). Pasirinkęs, bet kurį iš operatorių vartotojas gali

sužinoti tiekėjo siūlomus planus (*18 pav.*), o pasirinkęs ir pageidaujamą planą pamatyti jo tarifus. Programoje yra galimybė šalinti nebeegzistuojančius planus, taip pat koreguoti esamus (*19 pav.*).



17 pav. Tiekėjų sąrašas

18 pav. Planų sąrašas



19 pav. Galimybė plano duomenis trinti ir koreguoti

- **Planai**

Pasirinkus šį menu punktą, bus atvaizduoti visi DB esantys Tiekėjai ir planai (*20 pav.*)

20 pav. Planų sąrašas

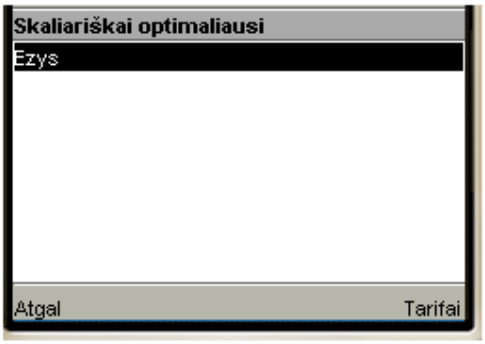
Pažymėkite laukelius šalia esančių planų, kurie vienaip ar kitaip yra patraukę jūsų dėmesį, pasirinkite metodą pagal kurį norite ieškoti optimalaus plano.

Pasirinkus *vektorinį* metodą iš visų planų bus atmesti neracionalūs planai, o atvaizduoti tik planai, kurie atrinkti pagal Pareto Optimaluma. Vartotojui bus pateikti tik atrinkti planai(*21 pav.*), greičiausiai po skaičiavimų liks tik keletas planų, kas vartotojui tikrai turėtų pagelbėti atliekant galutinį apsisprendimą. Telefone bus atvaizduotas atrinktų planų sąrašas, o pasirinkus dominantį planą galima peržiūrėti plano tarifus.

21 pav. Vektorinės optimizacijos rezultatas

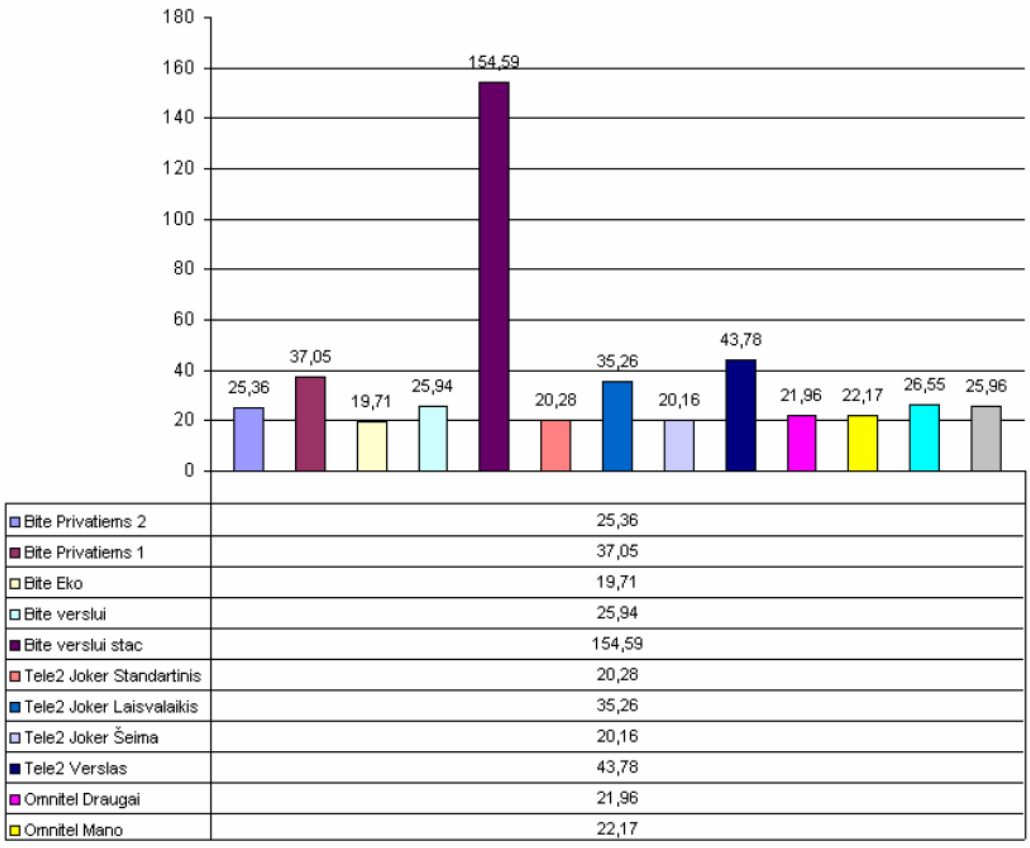
Pasirinkus *skaliarinį* metodą emuliatoriaus ekrane bus pavaizduota svorių įvedimo forma (*22pav.*), kurioje kiekvienam mobiliojo mokėjimo plano parametrui, galima įvesti svorį. Šiuo atveju svoriu mes nusakome kiekvieno parametro svarbumą atskiram vartotojui. Pavyzdžiui, tarkime vartotojas dažniausiai skambina į kitus tinklus, o SMS (trumpųjų žinučių pranešimų) visai nerašo, todėl jam parametro „Skambučiai į kitus tinklus“ svarbumas yra didesnis negu parametro„SMS“. Tuomet vartotojas turėtų parametrui „Skambučiai į kitus tinklus“ užduoti didesnį svorį, negu pavyzdžiui parametrui „SMS į kitus tinklus“. Pasirinkus mygtuką „Skaičiuoti“ pagal formulę bus išrinktas planas atitinkantis vartotojo pageidavimus (*23 pav.*).

22 pav. Svorių įvedimo langas



23 pav. Skaliarinės optimizacijos rezultatas

Kiekvieną kartą suvedus svorius, pagal formulę apskaičiuojamos skaliarinės reikšmės kiekvienam planui. Sprendinių bus mažiausia reikšmė. Kitais atvejais gali būti išrinkti ir keli planai, bet tai atsitiks tuo atveju kai skaliarinės reikšmės sutaps. Atvaizduokime reikšmės kitimą kiekvienam planui, jeigu kiekvienam parametrui užduosime svorį 1.0, o parametrui „Skambučiai į kitus tinklus“ svorį 5.0 (*24 pav.*):



24 pav. Skaliarinės reikšmės kitimo grafikas

Galima daryti prielaidą, kad minimali reikšmė – 19.71, esant tokiai svorių kombinacijai vartotojui geriausiai tinkamas mokėjimo planas – Bite Eko.

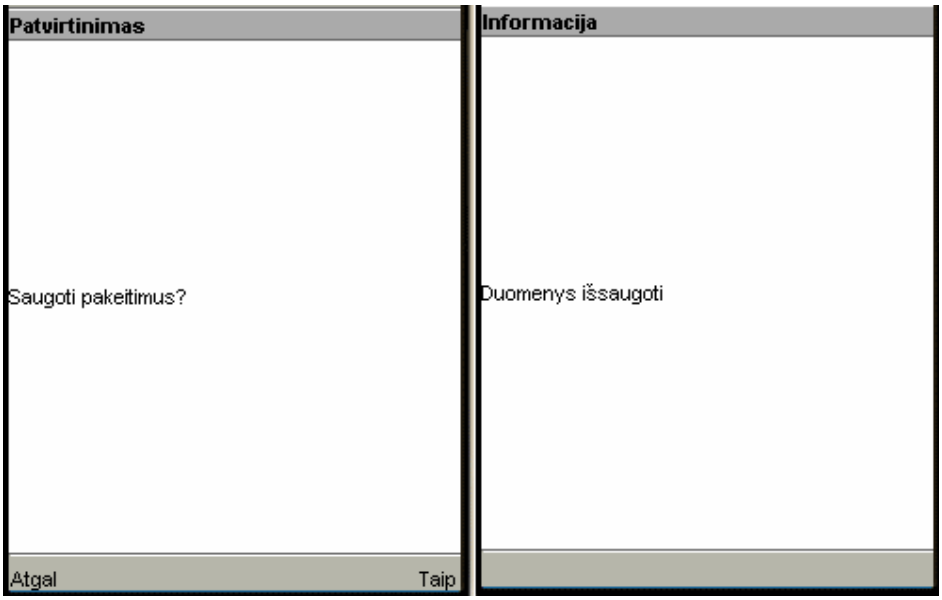
- **Naujas**

Kaip žinome mobiliojo ryšio operatoriai nesnaudžia vietoje, siekdami pralenkti konkurentus jie kuria naujus planus. Numatydamas naujų planų atėjimą į rinką, į savo programą įdiegiau galimybę ją papildyti naujais planais, tam **Menu** jums reikia pasirinkti punktą **Naujas.** Pasirinkus šį punktą, programa atidaro langą (*25 pav.*), kuriame turite suvesti naujo plano duomenis, ir paspausti mygtuką **Saugoti.**

25 pav. Duomenų papildymo forma

- **Saugoti**

Jei tarp planų atlikote pakeitimus (papildėte naujais, ištrynėte nebeegzistuojančius, koregavote tarifus) ir norite juos išsaugoti ateičiai, pasirinkite punktą – **Saugoti**. Jums tereikės patvirtinti, kad tikrai norite išsaugoti pakeitimus, apie išsaugojimą jūs taip pat būsite informuotas (26 pav.).



26 pav. Duomenų išsaugojimas

- **Pagalba –** trumpa pagalba vartotojui.
- **Baigti –** baigti programos darbą.

5.8. Programos testavimas mobiliųjų įrenginių aplinkose

Apžvelgus programą universaliame emuliatoriuje, išbandykime programą kelėtame tikrų įrenginių.

- Motorola V360
- Nokia 6230
- Sony Ericsson K750
- Siemens SK65
- Sony Ericsson T630

Pateikti įrenginių modeliai palaiko MIDP 2.0 ir CLDC 1.1, išskyrus T630. Šių telefonų specifikacijos:

- Motorola V360
    - Ekrano skiriamoji geba: 176x220
    - Spalvos: 18 bitu (262,144 spalvu)
    - Maksimalus 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 bitu (65,536 spalvu)
  - 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 bitu (262,144 spalvu)
  - 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 bitu (65,536 spalvu)
  - 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.

- Sony Ericsson T630
  - Ekrano skiriamoji geba: 128x160
  - Spalvos: 16 bitu (65,536 spalvu)
  - Saugomos atminties dydis: 2 Mb
  - Palaikomos technologijos: MIDP v1.0 ir CLDC v1.0

Telefonų emuliatoriai parsisiųsti iš gamintojo tinklapių. Emuliatoriams naudotos programos:

- 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

Kaip matome (*24 pav*.) vienintelis „Sony Ericsson“ atvaizdavo meniu gražiai ir taisyklingai. Realiai „Nokia“ telefonas taip pat susitvarkė su šia užduotimi tačiau emuliatorius neatpažino lietuviškų raidžių, tai greičiausiai įdiegtos koduotos trūkumas, taip pat keliose vietose buvo paslėptos *kelios* komandos, kas privedė prie to, kad pradžioje reikėjo įeiti i "*Funkcijos*" meniu ir tik po to paspausti pasirinkti norimą funkciją. Tai yra gana nepatogu vartotojo atžvilgiu. „Motorola“ ir „Siemens“ emuliatorių rezultatai panašūs, nors manyčiau „Siemens“ ekrane, meniu atrodo patraukliau negu „Motorola“. Šioje situacijoje galima išskirti du lyderius: besąlygiškai pirmas būtų „Sony Ericsson“ K750, o šalia jo „Siemens“ SK65 bei „Motorola“. Navigacijos mygtukai patogiausi pasirodė Sony Ericsson telefone, bet tai daugiau skonio reikalas.

24 pav. Pagrindinis menu skirtinguose emuliatoriuose

Akivaizdžiai matosi (*25 pav.*), kad kiekvienas telefonas skirtingai ir savaip atvaizdavo mobiliųjų mokėjimo planų informaciją. Galima pastebėti, kad Motorola gana neblogai susitvarkė su šia užduotimi. Jos emuliatorius duomenis atvaizdavo

taisyklingai, jų daugiausia tilpo ir ekrane. Lyginant Sony Ericsson su Motorola, dėl vienodos ekrano skiriamosios gebos 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“ pastebėjau, kad ne visai tikslus duomenų atvaizdavimas susijęs su ekrano skiriamąją geba. „Nokia“ skiriamoji geba 128x128, o „Siemens“ 132x176. Be to „Nokia“ skiria ekrano piešimui tik 128x96 šios gebos. Darome išvadą, kad nepaisant didesnės ekrano skiriamosios gebos „Siemens“ žymiai blogiau susidorojo su šia užduotimi, ekrane atvaizduodamas neaiškiai išsibarsčiusius simbolius. Galima teigti, kad programoje aprašytą „\n“ simbolį, kuris turėtų pernešti žymeklį į kitą eilutę, Siemens emuliatorius nepastebėjo.

25 pav. Informacijos atvaizdavimas

Panagrinėkime atliktus skaičiavimus. Greičiausiai, pagal Pareto optimumą, mobiliuosius mokėjimo planus atrinko „Motorola“ ir „Sony Ericsson“, labai nedaug atsiliko „Nokia“. „Siemens“ palyginus su kitais skaičiavo apytikriai 5 sekundėmis ilgiau daugiau, nors tai galėjo būti susiję su „Siemens“ programine įranga, bei emuliatoriumi. Vektorinį skaičiavimą su svoriais, uždavus kiekvienam parametrui svorį 20, visi telefonai skaičiavimus atliko apylygiai.

Apibendrinant aukščiau paminėtus palyginimus, bei telefonų modelius galima būtų išskirti du telefonus: „Motorola“ bei „Sony Ericsson“. Kurie gražiai atvaizdavo visą programą, nors ji 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 ši bendrovė aktyviai bendradarbiauja su Sun Microsystem (J2ME kurėja) ir todėl programinės priemonės atrodo labai panašiai kaip ir teikiamo J2ME emuliatoriaus.

Todėl, kuriant J2ME programa „Sony Ericsson“ telefonui, galima testuoti standartiniame emuliatoriuje. Telefonų gamintojų „Nokia“, „Siemens“, pasirodžiusių prasčiau, galima pažymėti tai, kad tai gali būti susiję su mažesne skiriamąją geba bei programine įranga, kurią teikia šių telefonų gamintojai.

Kol kas bandymuose nepaminėjau tik „Sony Ericsson“ produkto T630, be abejo, panašių telefonų modelių yra daugiau, tačiau kadangi jis yra vienas populiariausių todėl pasirinkau būtent jį. Dėl to senesnio MIDP v1.0 profilio palaikymo man taip ir nepavyko jame paleisti šios programos, kompiliuojant programos kodą, klaidų pranešimuose buvo galime įžvelgti, kad kompiliatorius neatpažįsta versijoje MIDP v2.0 palaikančių interfeiso produktų. Taigi greičiausiai senuosiuose telefonuose, palaikančiuose senesnes versijas MIDP ir CLDC šios programos neįsidiegsite.

5.9. Eksperimentų išvados

Darbo eigoje pastebėjau, kad telefonų emuliatoriai, yra naudinga priemonė, išbandyti pirminius programos variantus. Naudodami emuliatorių ne tik sutaupysite laiko, kurį sugaištumėte programą diegdami i mobilųjį įrenginį, bet ir pinigų, nereikės pirkti telefonų.

Tarkim pats, asmeniškai, naudoju „NOKIA 6230“ modelį, telefono ir emuliatoriaus darbas visiškai nesiskyrė, o išbandyti programą emuliatoriuje buvo daug lengviau, nes kad įdiegčiau programą į savo mobilųjį telefoną man prireikė ir papildomos programinės ir techninės (Bluetooth arba kabelio) įrangos.

Vertėtų atkreipti dėmesį į tai, kad rašydamas programą panaudojau keletą grafinių elementų, (sistemos krovimo indikatorius, kiti informuojantys pranešimai). Tačiau sistemoje pradėjus naudoti RMS, mano programos padailinimai dingo. Naršydamas interneto forumuose taip pat neradau jokios informacijos ar tai gali būti papildomai naudojamų priemonių pasekmė.

Reikia pastebėti, kad kuriant programa, reikia būtina 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.

6. Išvados

1. Šiame darbe išanalizuotas ir ištirtas papildomas Informacinių technologijų (IT) panaudojimas ir reikalingumas šiuolaikiniuose mobiliuosiuose įrenginiuose, aptarti pagrindiniai šios srities aspektai.
2. Darbe tirta mobiliųjų įrenginių programinės įrangos priemonės ir jų galimybės apdoroti optimizacinius metodus.
3. Eksperimentinėje dalyje išnagrinėtas ir pritaikytas Pareto optimizacinis modelis.
4. Įgyvendintas darbe iškeltas tikslas – sukurti Pareto optimalumo metodu paremtą sistemą, kuri būtų skirta sutrumpinti, palengvinti ir pagreitinti vartotojo apsisprendimą pasirenkant mobiliojo ryšio tiekėją ir planą.
5. Atlikus priemonių analizę, nustatyta, kad šiuo metu universaliausia mobiliesiems įrenginiams skirta programavimo priemonė yra J2ME, ja parašytą programos kodą galima pritaikyti daugeliui įrenginių.
6. Programoje panaudota įrašų valdymo sistema, padidina programos galimybes redaguoti duomenis apie planus.
7. Programa buvo kurta testavimui pasitelkiant standartinį „DefaultColorPhone“ Java 2 Wireless Toolkit emuliatorių ir teoriškai ji turėtų gerai veikti kiekviename telefono modelyje, tačiau išanalizavus programos veikimą kituose telefonų emuliatoriuose, kai kur būta grafinių neatitikimų. Nepaisant to, optimalūs mobilieji mokėjimo planai, kiekviename emuliatoriuje, buvo išrinkti teisingai. Grafinius neatitikimus galima paaiškinti tuo, kad kuriant programą, reikėtu ją kurti konkrečiam telefono modeliui, atsižvelgiant į jo techninius parametrus.

7. Literatūra

- Žilinskas. Matematinis programavimas. Kaunas: Vytauto Didžiojo universiteto leidykla, 2000, 227 p.
- А. Жилинскас, В. Шалтянис. Поиск оптимума. Москва: Наука, 1989, 128 с.
- Žilinskas. Naujieji projektavimo metodai. Vilnius: Mokslas, 1990, 109 p.
- С.Г. Горнаков. Программирование мобильных телефонов на Java 2 Micro Edition. Москва: ДМК Пресс, 2004, 336 с.
- Martin J. Wells. J2ME ™ Game Programming. Premier Press, 2004, 803 p.
- Habil. dr. prof. Jonas Mockaus asmeninis interneto puslapis, skirtas optimizicijai. http://soften.ktu.lt/~mockus
- Sun Microsystems Java 2 Wireless toolkit. http://java.sun.com/products/j2mewtoolkit/
- Сайт потребителей и разработчиков Java. http://www.juga.ru/
- Naujoji komunikacija. http://www.nktv.lt/
- Сайт потребителей и разработчиков Java. http://www.javable.com/
- NetBeans. http://www.netbeans.org/kb/41/mobility.html
- Omnitel. http://www.omnitel.lt/
- Tele2. http://www.tele2.lt/
- Bitė. http://www.bite.lt/
- http://www.soften.ktu.lt/~mockus/
- http://java.sun.com/j2se/1.3/docs/guide/security/spec/security-spec.doc
- http://java.sun.com/aboutJava/communityprocess/final/jsr030/CLDCSpecification1.0
- http://java.sun.com/products/cldc/wp/KVMwp.pdf
- http://www.javasoft.com/products/jdk/1.2/docs/api/index.html
- http://java.sun.com/aboutJava/communityprocess/jcp2.html
- http://developers.sun.com/techtopics/mobility/device/device
- http://developers.sun.com/techtopics/mobility/midp/chapters/j2meegiguere_pg/Chapter1.pdf

Summary

The Research of Optimization methods using Mobile Devices

The usage of mobile devices is increasing every day. The customers' choice depends on what features does one or the other device supports, how does it look like or other specification that he needs. Most of the new models support J2ME (Java 2 Micro Edition), allow user to extend his mobile devices features. Integrated J2ME, MIDO and CLDC modules – that's all you need if you want to add new software into your device.

The work goal is to study fundamental optimization models, their analyze methods, algorithms to solve optimization tasks use them in mobile devices. The task is using J2ME write an application which base is Pareth Optimum method for mobile device. During power-stroke assimilate MIDP v2.0 and CLDC v1.1, go into the optimization theory knowledge, learn to align mathematics and informatics methods solving tasks, study optimization problems classes and their methods, score skills rating practical tasks their complexity choosing best methods to solve them.

8. Priedai

- **Mobiliųjų planų išrinkimas skaliariškai**

```
protected void DisplayScalar() {

        double skaliarai[] = new double[displayedPlans];
        int i, j;

        // find scalar plan values
        for ( i = 0; i < displayedPlans; i++ ) {
            skaliarai[i] = 0;
            for ( j = 0; j < parametru_sk; j++ )
                skaliarai[i] +=
parametrai[planDisplayIndex[i]][j]*daugikliai[j];
        }

        // find best plan value
        double geriausias = skaliarai[0];
        for( i = 1; i < displayedPlans; i++ )
            if (skaliarai[i] < geriausias)
                geriausias = skaliarai[i]; // geriausias -
maziausias

        // remove not the best plans
        for( i = displayedPlans-1; i >= 0; i-- ) {
            if ( skaliarai[i] == geriausias )
                continue;

            for (j=i; j < displayedPlans-1; j++)
                planDisplayIndex[j] = planDisplayIndex[j+1];
            displayedPlans--;
        }

    } /* */
```

- **Mobiliųjų planų išrinkimas vektoriškai**

```
protected void DisplayVect() {

        int j, i, k, l, nr = 0, pr = 0, pab = parametru_sk, temp;
        boolean geras = true, param_geriau = false, param_blogiau =
false;
        int map[] = new int[parametru_sk]; // indexes for parameters
        for ( i = 0; i < parametru_sk; i++)     {
            map[i] = i;
        }

        // bubble sort parameters by importance, from least to most
        for ( i = 1; i < parametru_sk - 1; i++) {
            for ( j = 0; j < parametru_sk - i; j++) {
                if ( daugikliai[map[j]] > daugikliai[map[j + 1]]) {
                    temp = map[j];
                    map[j] = map[j + 1];
                    map[j + 1] = temp;
            }
            }
        }
```

```
        // parameters with negative or zero weights are omitted
        while ( nr < parametru_sk && daugikliai[map[nr]] <= 0)
            nr++;

        // main cycle - eliminates not the best
        while ( (nr < parametru_sk) && (displayedPlans > 1) ) {
            /// parameters with the same weight are processed at the
same time
            /// determine range of such parameters
            pr = pab = nr;
            while ( (nr + 1 < parametru_sk) && (daugikliai[map[nr]]
== daugikliai[map[nr + 1]])) {
                nr++;
                pab = nr;
            }

            // eliminate bad plans for given range
            for ( i = displayedPlans-1; i >= 0; i-- ) {
                // checking if plan i is good
                geras = true;
                for ( k = 0; (geras == true) && (k < displayedPlans);
k++) {
                    if ( k == i ) // don't check a plan against
itself
                        continue;

                    // check if plan i is good in comparison to plan
k
                    param_geriau = false;
                    param_blogiau = false;
                    for ( l = pr; (param_geriau == false) && (l <=
pab); l++) {
                        if ( parametrai[planDisplayIndex[i]][map[l]]
> parametrai[planDisplayIndex[k]][map[l]] )
                            param_blogiau = true; // kuo mazesnis,
tuo geresnis
                        if ( parametrai[planDisplayIndex[i]][map[l]]
< parametrai[planDisplayIndex[k]][map[l]])
                            param_geriau = true;
                    }

                    // if plan i is totally worse than k, it is to be
removed
                    if ( !param_geriau && param_blogiau)
                        geras = false;
                }

                // removing plan i, if not good
                if (!geras) {
                    for (k=i; k < displayedPlans-1; k++)
                        planDisplayIndex[k] = planDisplayIndex[k+1];
                    displayedPlans--;
                }
            }

            // and, finally
            nr++;
        }

    } /* */
```