KAUNO TECHNOLOGIJOS UNIVERSITETAS
Informatikos fakultetas

TVARKARAŠČIO OPTIMIZAVIMAS

Modulio "Optimizavimo metodai" kursinis darbas

Atliko : IFM 5/1 gr. studentas
Vytautas Germanavičius

Priėmė :
prof. J.Mockus

KAUNAS 2000

Turinys

1. Užduotis

Pertvarkyti duotą mokyklos pamokų tvarkaraštį minimizuojant mokytojų "langų" skaičių. Gautas tvarkaraštis turi tenkinti reikalavimus pateiktus skyriuje "Reikalavimai"(žr. žemiau).

Darbas turi būti atliktas, kaip GMJ programos (http://www.soften.ktu.lt/~mockus/) modulis. Modulis turi užduoti funkciją optimizavimui, šiuo atveju – funkcija yra mokytojų langų priklausomybė nuo mokytojo praleidimo tikimybės tvarkaraščio pertvarkymo metu.

1.1. Reikalavimai

Tvarkaraštis turi tenkinti tokius reikalavimus :

- Mokiniai negali turėti "langų".
- Mokiniai vienu metu gali turėti tik vieną pamoką.
- Mokytojai vienu metu gali turėti tik vieną pamoką.
- Negali būti dviejų iš eilės to pačio dalyko pamokų.
- Pamokų skaičius per dieną yra ribotas.

1.2. Žodynas

Laisvas laikas - pamoka(os), kai dėstytojas yra laisvas.
Langas – Laisvas laikas tarp dviejų pamokų, per kurias mokytojas yra užimtas.
Einamas mokytojas – mokytojas, kurio tvarkarašti yra nagrinėjamas šiuo metu.
Einamas langas – einamojo mokytojo langas, kuri bandoma panaikinti.

2. Algoritmas

1. Užpildomas pradinis tvarkaraštis.
2. Nustatomas iteracijų skaičius.

Kiekvienai iteracijai :

3. Išrenkamas einamasis mokytojas (pirmasis)

Kiekvienam mokytojui :

4. Sugeneruoti atsitiktini skaičių $\xi \in [0,1]$.
5. Jei $\xi < x$, pereiti prie kito mokytojo. Čia x – konstanta.
6. Išrinkti einamojo mokytojo langą, kurį bus bandoma panaikinti. Šioje realizacijoje tai pats pirmas langas einamojo mokytojo tvarkaraštyje.
7. Išrinkti pamoką pakeitimui iš kiekvienos dienos pirmų ir paskutinių pamokų aibės. Ją įstačius lango vietoje turi būti tenkinami skyriuje "Reikalavimai" išdėstyti reikalavimai. Jei tokiu nėra, pereinama prie kito mokytojo.
8. Rasti mokytoją, kuris turi pamoka, einamo mokytojo lango metu, klasei, kuriai turi pamoka einamasis mokytojas, išrinktosios pamokos metu. Jei tokio nėra atlikti pakeitimą einamojo mokytojo tvarkaraštyje.
9. Patikrinti, ar rastajam mokytojui sukeitus pamokas einamojo lango ir išrinktosios pamokos metu, bus tenkinami skyriuje "Reikalavimai" išdėstyti reikalavimai. Jei netenkinami – vykdyti 7 punktą.
10. Atliekamas pakeitimas einamo ir rastojo mokytoju tvarkaraščiuose.
11. Palyginamas langų skaičius ankstesniame ir gautajame tvarkaraščiuose. Jei gautasis turi daugiau langų, atšaukti pakeitimus.

Algoritmas paruoštas sekant algoritmu pateiktu literatūroje [1] psl. 294-295.

2.1. Algoritmo įvertinimas

Algoritmas yra labai paprastas. Tačiau jame yra trūkumų : Uždarant langą vienam, atsidaro langas kitam mokytojui. Langai panaikinimui imami iš eilės. Pamokos lango panaikinimui irgi imamos iš eilės. Taigi gali susidaryti tokia situacija, kad bus be galo naikinami tik pirmos dienos langai : vienam panaikinus pirmą langą atsidaro pirmas langas kitam, uždarius šiam – atsidaro pirmajam.

3. Realizacija

Užduoties realizacija yra iš dviejų dalių : GMJ modulio ir atskiros tvarkaraščio pertvarkymo programos. Iš esmes, tai viena programa, tik turinti du vartotojo interfeisus : viena GMJ programai, kita atskirai programai. Tvarkaraščio sugojimo, pertvarkymo ir pan. objektai yra vienodi abiem atvejais.

3.1. Duomenų struktūra

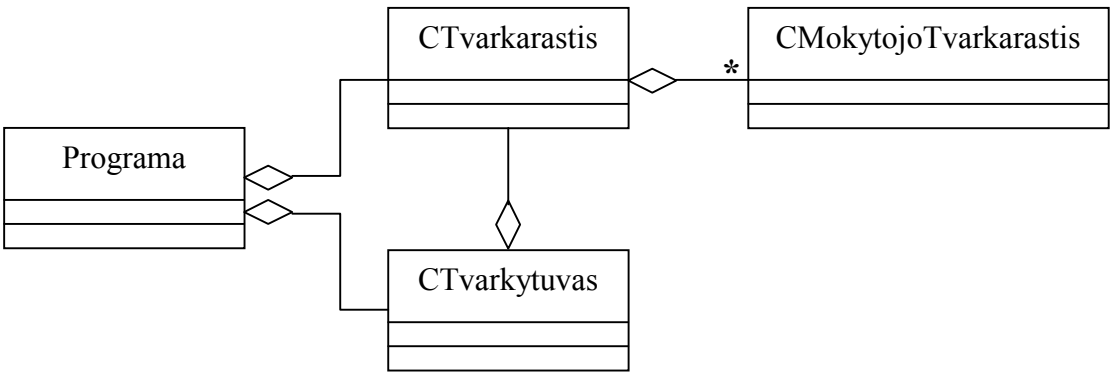
Duomenys – bendras tvarkaraštis – programoje saugomas, kaip mokytojų tvarkaraščių masyvas.

Mokytojo tvarkaraštis – tai mokytojo vardas-pavardė, dėstomas dalykas ir pamokų sąrašas, masyvo formoje, kurio kiekvienas elementas rodo, kokiai klasei turi pamoką mokytojas arba yra laisvas; masyvo ilgis lygus pamokų skaičiui per savaitę.

Tvarkaraštyje laisvas mokytojo laikas žymimas "**---**" , mokytojo langas – "**-W-**", kiti elementai užpildomi klasių, kurioms mokytojas turi pamokas, pavadinimais.

3.2. Klasių diagramos

3.2.1. Bendroji dalis



Klasė Programa

Klasė vartotojo interfeisui sukurti. GMJ atvejui, tai - klasė CTvarkarastisTask, atskiros programos – CPrograma.

Klasė CTvarkarastis

Klasė skirta bendro tvarkaraščio saugojimui, jo tvarkymui ir įvertimui.

Kintamieji :

public int kiekMokytoju; // mokytoju kiekis tvarkaraštyje
public CMokytojoTvarkarastis[] mokytojas; //mokytoju tvarkaraščiai

Metodai :

public CTvarkarastis(int _kiekMokytoju)
Klasės konstruktorius. Sukuria tuščia tvarkaraštį, kai mokytojų skaičius - *_kiekMokytoju*.

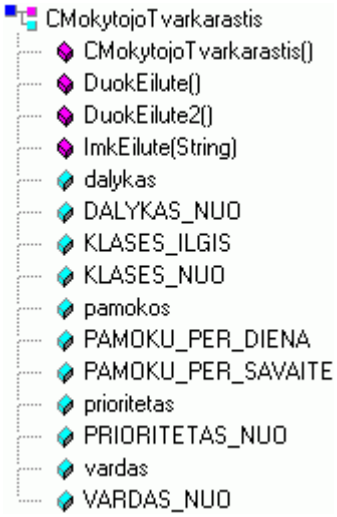
public int SkaiciuokLangus(int kam)
Skaiciuoja parametru *kam* nurodyto mokytojo langu skaičių. Jei bus naudojami prioritetai, rezultatą reikės padauginti iš prioriteto.
Algoritmas. Skaičiuojami tie pamokų masyvo nariai, kurie lygūs "-w-".

public void ZymekLangus(int kam)
Atskiria langus nuo laisvo laiko parametru *kam* nurodytam mokytojui:

- langus zymi "**-W-**"
- laisva laika zymi "**---**"

Algoritmas. Pirmiausia visi langai ir laisvas laikas pažymimi **"---"**. Po to imama po vieną diena ir surandamos pirmos ir paskutinės pamokos. Tarp jų esančius laisvus laikus, pažymi kaip langus – **"-W-"**.

Klasė CMokytojoTvarkarastis



Klasė skirta atskiro mokytojo tvarkaraščio saugojimui.

Konstantos :

Konstantos duomenų įvedimui:
public static final int VARDAS_NUO = 0;
Pozicija, nuo kurios prasideda vardas-pavardė paduodamoje duomenų eilutėje.

public static final int DALYKAS_NUO = 22;

Pozicija, nuo kurios prasideda dėstomo dalykopavadinimas paduodamoje duomenų eilutėje.

public static final int KLASES_NUO = 37;
Pozicija, nuo kurios prasideda mokytojo tvarkaraštis paduodamoje duomenų eilutėje.

public static final int KLASES_ILGIS = 3;
Klasės pavadinimui išskirtų simbolių kiekis paduodamoje duomenų eilutėje.

Tvarkaraščio konstantos :
public static final int PAMOKU_PER_DIENA = 7;
Maksimalus pamokų skaičius per dieną.

public static final int PAMOKU_PER_SAVAITE = PAMOKU_PER_DIENA*5;
Maksimalus pamokų skaičius per penkių dienų darbo savaitę.

Kintamieji :

```
public String[] pamokos;      // mokytojo pamokų masyvas;
public String dalykas  = "";  // mokytojo dėstomo dalyko pavadinimas;
public String vardas   = "";  // mokytojo vardas-pavardė;
```

Metodai :

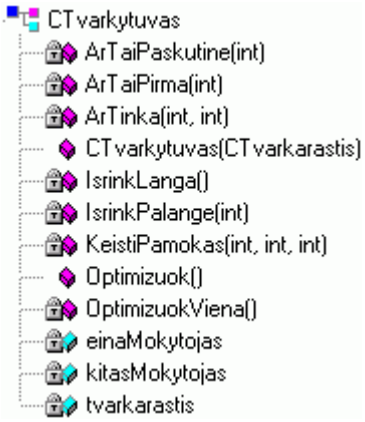
public CMokytojoTvarkarastis ()
Klasės konstruktorius. Sukuria tuščią mokytojo tvarkaraštį.

public void ImkEilute(String eilute)
Suformuoja mokytojo duomenis iš eilutės *eilute*.
Algoritmas. Suskaidyti *eilute* į dalis kurias iš eilės priskirti *vardas*, *dalykas,* atitinkamiems *pamokos* elementams.

public String DuokEilute()
Perveda duomenis į eilutės (String) formą. Vizualiai aiškesnis variantas. Naudojamas testavimui arba alternatyviam duomenų pateikimui. Netinka pakartotiniam padavimui kaip duomenis.

public String DuokEilute2()
Analogas *public String DuokEilute()***.** Perveda duomenis į eilutės (String) formą. Rezultatas tinka pakartotiniam duomenų įvedimui.

Klasė CTvarkytuvas



Klasė skirta tvarkaraščių optimizavimui.

Kintamieji :

private int einaMokytojas = -1;
Einamas mokytojas

private int kitasMokytojas = -1;
Kitas mokytojas, su kuriuo einamas keisis pakaitomis

private CTvarkarastis tvarkarastis;
Optimizuojamas tvarkaraštis

Metodai :

public CTvarkytuvas(CTvarkarastis _tvarkarastis)
Klasės konstruktorius. Sukuria tvarkytoją tvarkaraščiui, užduotam parametru *_tvarkarastis*. Atlieka pradinį tvarkaraščio sužymėjimą.

private boolean ArTaiPaskutine(int kas)
Funkcija pasako, ar parametras *kas* rodo paskutinę pamoką tą dieną.
Algoritmas. *kas* dalinti iš pamokų skaičiaus per dieną ir imti liekaną, jei liekna lygi pamokų skaičiaus per dieną –1, tai *kas* rodo paskutinę pamoką.

private boolean ArTaiPirma(int kas)
Funkcija pasako, ar parametras *kas* rodo pirmąją pamoką tą dieną.
Algoritmas. *kas* dalinti iš pamokų skaičiaus per dieną ir imti liekaną, jei liekna lygi 0, tai *kas* rodo pirmą pamoką.

private boolean ArTinka(int kam, int kas)
Parametras *kas* žymi išrinktą pamoką pakeitimui iš kiekvienos dienos pirmų ir paskutinių pamokų aibės. Ją įstačius lango vietoje, jį žymi parametras *kas*, turi būti tenkinami skyriuje “Reikalavimai” išdėstyti reikalavimai. Funkcija atlieka šį tikrinimą.
Taip pat randa mokytoją, kuris turi pamoka, einamo mokytojo lango metu, klasei, kuriai turi pamoka einamasis mokytojas, išrinktosios pamokos metu. Patikrina, ar rastajam mokytojui sukeitus pamokas einamojo lango ir išrinktosios pamokos metu, bus tenkinami skyriuje “Reikalavimai” išdėstyti reikalavimai.

Rezultatas :

- true – jei keitimas neprieštarauja skyriuje “Reikalavimai” išdėstytiems reikalavimams.
- false – priešingu atveju.

Algoritmas. Keisti negalima, jei :

1. dvi is eiles tos pacios paskaitos
2. vienai klasei dvi pamokos

Keisti galima tik su ta, kuri nelygi prieš ir po einančiai pamokai.

Kitam mokytojui turi būti langas keičiamosios pamokos metu, arba tokia pamoka kuri nelygi prieš ir po einamo lango einančiai pamokai.

private void KeistiPamokas(int kam, int ka, int suKuo)

Sukeičia mokytojui, nurodytam parametru *kam*, pamokas nurodytas parametrais *ka* ir *suKuo*.

private int IsrinkLanga()

Išrenka langą iš einamo mokytojo tvarkaraščio, kuri reikia optimizuoti. Imamas pats pirmas langas. Gražina –1, jei nėra langų.

private int IsrinkPalange(int kuriai)

Išrinka pamoką pakeitimui iš kiekvienos dienos pirmų ir paskutinių pamokų aibės. Ją įstačius lango vietoje turi būti tenkinami skyriuje “Reikalavimai” išdėstyti reikalavimai. Jei tokių neranda, gražina –1.

Randa mokytoją, kuris turi pamoka, einamo mokytojo lango metu, klasei, kuriai turi pamoka einamasis mokytojas, išrinktosios pamokos metu. Jei tokio nėra atlikti pakeitimą einamojo mokytojo tvarkaraštyje.

Patikrina, ar rastajam mokytojui sukeitus pamokas einamojo lango ir išrinktosios pamokos metu, bus tenkinami skyriuje “Reikalavimai” išdėstyti reikalavimai.

public void Optimizuok()

Nustato optimizavimo iteracijos numerį ir einamą mokytoją.

private void OptimizuokViena()

Pagrindinis klasės metodas. Jame atliekamas visas optimizavimo darbas einamam mokytojui..

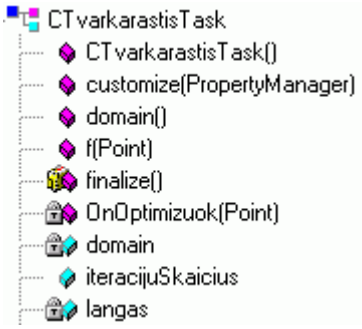
Algoritmas.

1. Einamam mokytojui sužymėti langus
2. Išrinkti langą, kuri optimizuosime
3. Išrinkti su kokia pamoka keisti, kartu tikrinant su kuo negalima keisti.
4. Jei neradom keitimo, pereiti prie kito mokytojo
5. Jei išrinkta klasė laisva lango metu - keisti be jokiu tikrinimu ir pereiti prie kito mokytojo.
6. Gauti dabartinio tvarkaraščio įvertinimą
7. Atlikti nustatytą pakeitimą.
8. Gauti ivertinimus po keitimo
9. Palyginti ankstesnį tvarkaraštį su tvarkaraščiu po keitimo. Jei rezultatas blogesnis – atkeisti

3.2.2. GMJ modulis

Apie pačią GMJ programą sunku ką nors pasakyti, nes ji praktiškai nedokumentuota.(žr. [2]). Yra tik aprašas kokios klasės turi būti modulyje. Todėl ir aš nepateikiu GMJ modulio klasių diagramos.

Klasė CTvarkarastisTask implements Task



Klasė skirta užduoti funkciją optimizavimui .

Kintamieji :

private CTvarkarastisDomain domain
vartotojo interfeisas duomenų įvedimui.

public int iteracijuSkaicius = 1;
Iteracijų skaičius, per kurį atliekamas tvarkaraščio pertvarkymas.

private static CLangas langas;
Langas tvarkaraščio įvedimui.

Metodai :

public void customize (PropertyManager manager)
Sukuria interfeisą iteracijų skaičiui įvesti.

public Domain domain ()
Grąžina sukurtą interfeisą.

public CTvarkarastisTask()
Klasės konstruktorius. Sukuria duomenų įvedimo langą.

public double f (Point pt)
Funkcija optimizavimui. Blokuoja duomenų keitimo galimybę ir kreipiasi į OnOptimizuok.

private int OnOptimizuok(Point pt)
Metodas gauna mokytojo praleidimo tikimybę ir išduoda kiek langų pertvarkytame tvarkaraštyje

Klasė CTvarkarastisDomain extends Domain

Klasė skirta sukurti vartotojo interfeisą GMJ moduliui.

Kintamieji :

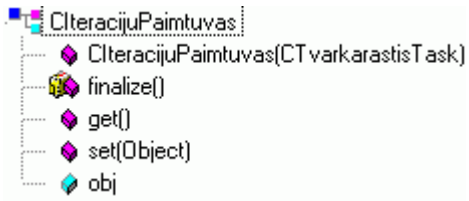
static final String []dimensions={"Tikimybė x1:"};

Metodai :

public String[] dimensions ()

CTvarkarastisDomain ()

Klasė CIteracijuPaimtuvas extends SimplePropertyProvider



Klasė skirta sukurti vartotojo interfeisą GMJ moduliui.

Kintamieji :

public CTvarkarastisTask obj;
Nurodo kokiam moduliui skirtas interfeisas.

Metodai :

public CIteracijuPaimtuvas (CTvarkarastisTask _obj)
Klasės konstruktorius.

public Object get ()
Išduoda kintamojo, su sieto su interfeiso objektu, reikšmę.

public void set (Object value) throws InvalidPropertyException
Nustato kintamojo, su sieto su interfeiso objektu, reikšmę.

4. Vartotojo vadovas

4.1. Duomenų pateikimas programai

Programai paduodami duomenys turi būti teisingi, tenkinti visus skyriuje "Reikalavimai" išdėstytus reikalavimus.

Duomenys programai paduodami paprastu tekstu eilutėmis, kur kiekviena eilutė yra vis kito mokytojo tvarkaraštis.

Mokytojo tvarkaraštis turi būti tokios struktūros :

<vardas-pavardė 22 simboliai><dalyko pavadinimas 15 simbolių><tvarkaraštis>

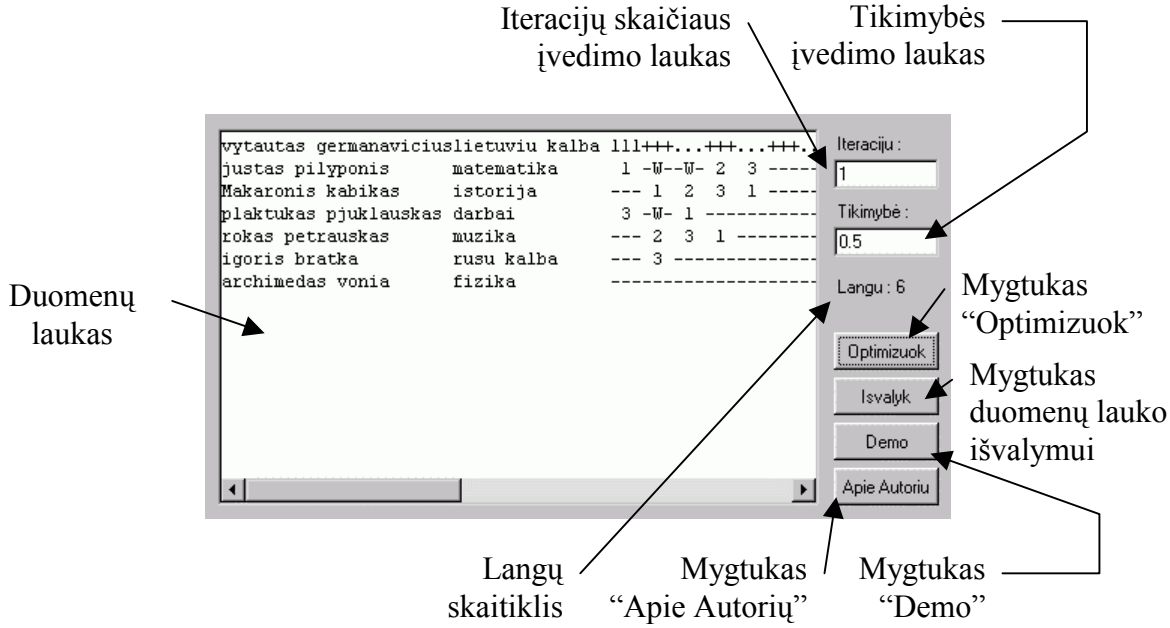
<*Tvarkaraštis*> turi būti tokios struktūros :

<*klasės pavadinimas 3 simboliai*>*X pamokų skaičius per savaitę*

<*klasės pavadinimas*> gali būti klasės pavadinimas arba simboliai "---", jei mokytojui laisvas laikas.

4.2. Atskira programa

4.2.1. Pagrindinis langas



Duomenų laukas – laukas duomenų įvedimui ir optimizavimo rezultatu ivedimui.

Iteracijų skaičiaus įvedimo laukas – laukas iteracijų skaičiaus įvedimui.

Tikimybės įvedimo laukas – laukas tikimybės, kad einamas mokytojas bus praleistas.

Langų skaitiklis – rodo paliktų langų skaičių po kiekvienos iteracijos.

Mygtukas "Optimizuok" – optimizuoja tvarkaraštį, užduotą duomenų lauke, per tiek iteracijų, kiek nurodyta iteracijų skaičiaus įvedimo lauke.

Mygtukas duomenų lauko išvalymui – išvalo duomenų lauką.

Mygtukas "Demo" – įveda į duomenų lauką demonstracinius duomenis.

Mygtukas "Apie Autorių" – pateikia informaciją apie autorių ir programą.

4.2.2. Rezultatų pateikimas

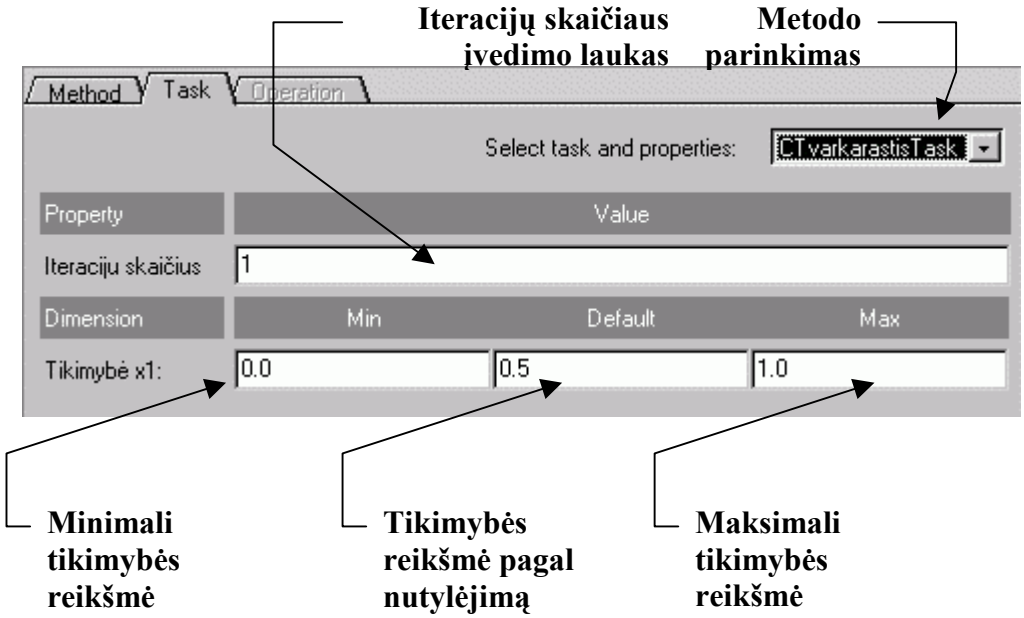
Rezultatai pateikiami tame pačiame duomenų įvedimo lange.

Pateikiamų rezultatų formatas toks pats kaip ir paduodamų duomenų, taigi gauti rezultatai vėl gali būti panaudoti kaip duomenys optimizavimo programai.

4.3. GMJ modulis

Pats GMJ aprašytas literatūroje [2], arba puslapyje internete http://www.soften.ktu.lt/~mockus. Čia pateikta tik tai kas susiję su tvarkaraščio uždaviniu.

4.3.1. Pagrindinis langas



Tikimybė – čia turima omenyje mokytojo praleidimo tikimybė pertvarkant tvarkaraštį.

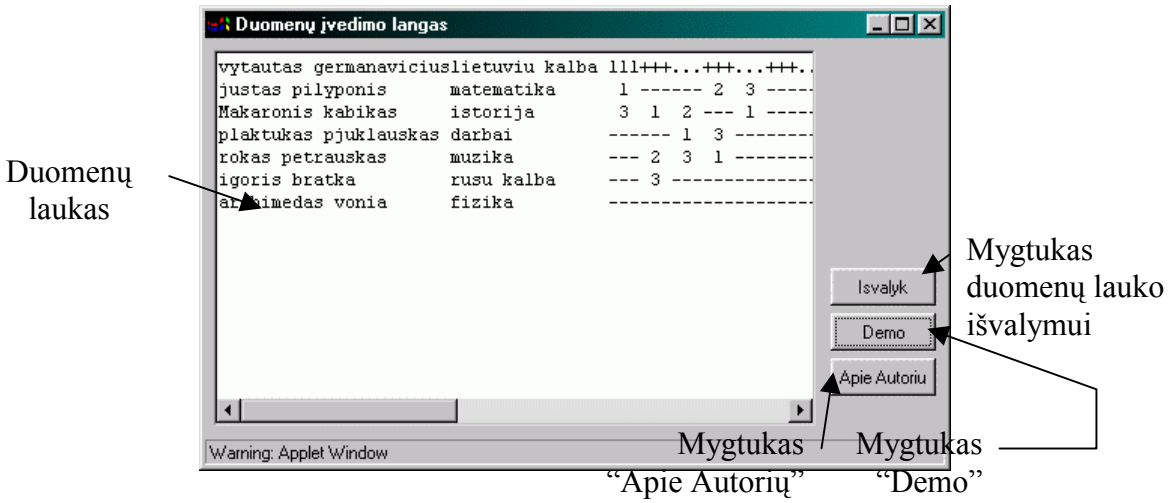
4.3.2. Duomenų įvedimo langas

Duomenų laukas – laukas duomenų įvedimui ir optimizavimo rezultatu ivedimui.

Mygtukas duomenų lauko išvalymui – išvalo duomenų lauką.

Mygtukas "Demo" – įveda į duomenų lauką demonstracinius duomenis.

Mygtukas "Apie Autorių" – pateikia informaciją apie autorių ir programą.



4.4. Darbo eiga.

Dėl operacinių sistemų specifikos darbo eiga su programa skiriasi skirtingose operacinėse sistemose.

4.4.1. Duomenų įvedimas

Tekstiniu redaktoriumi, arba duomenų įvedimo lange suvesti duomenis skyriuje "Duomenų pateikimas programai" aprašytu formatu. Naudojant tekstini redaktorių reikia pažymėti visus duomenis tekstiniame redaktoriuje, ***"Copy"*** komandos pagalba nukopijuoti, tada duomenų įvedimo lauke atlikti komanda ***"Paste"***.

Tiems kurie jau turi suvestu duomenis faile, arba Unix šeimos OS vartotojams rekomenduojama atsidaryti failą su antru naršyklės egzemplioriumi, tada pažymėti tekstą ir pasirinkti .naršyklės meniu punktą "**Copy**", tada pereiti į pirmąjį ir, nustačius kursorių duomenų įvedimo lauke, pasirinkti naršyklės meniu punktą "**Paste**".

Duomenys paduoti.

4.4.2. Rezultatų paėmimas

Dirbant su atskira programa, gauti rezultatai pateikiami duomenų įvedimo lauke. Norint juos išsaugoti, reikia pažymėti visus rezultatus duomenų įvedimo lauke, ***"Copy"*** komandos pagalba nukopijuoti, tada tekstiniame redaktoriuje atsidarius tuščią failą, atlikti komanda ***"Paste".*** Naująjį failą išsaugoti komanda ***"Save"***.

4.4.3. Patarimai pradedantiems vartotojams

Tekstiniai redaktoriai:

- Windows šeimos OS : notepad, word ir pan.;
- Unix šeimos OS : vi, pico ar pan.

Windows šeimos OS Copy/Paste komandas galima atlikti pasirinkus atitinkamai "Edit/Copy" / "Edit/Paste" meniu punktus arba klavišų kombinacijomis "Ctrl+c"/"Ctrl+v" arba "Ctrl+Ins"/"Shift+Ins".

Unix šeimos OS, neturi vieningos visom programoms "Copy/Paste" sistemos, t.y. kiekviena programa savaip realizuoja "Copy/Paste" komandas.

5. Literatūra

1. **J.Mockus.** A SET OF EXAMPLES OF GLOBAL AND DISCRETE OPTIMIZATION.Home Work for Graduate Students. Kluwer Academic Publishers. Boston / Dordrecht / London

2. **Modestas Grybauskas.** Global Minimizer for Java (GMJ) Version 1.1. Developer's and Power User's Guide. Kaunas. 1997