Realizace výpočetních experimentů a Analýza průběžných a finálních výsledků výpočetních experimentů – shrnutí za čtvrté monitorovací období

Tento dokument shrnuje dílčí dosažené výsledky pro dvě projektové aktivity, které spolu úzce souvisejí. První aktivita Realizace výpočetních experimentů je zajišťována hlavním přeshraničním partnerem Žilinskou univerzitou v Žilině a hlavním cílem této aktivity je realizovat optimalizační a simulační experimenty vedoucí k naplnění cílů projektu. Tato aktivita je potom provázána s druhou aktivitou Analýza průběžných a finálních výsledků výpočetních experimentů, která je zajišťována vedoucím partnerem VŠB – Technickou univerzitou Ostrava.

Tato aktivita byla ve sledovaném monitorovacím období ukončena, avšak s posunem vůči termínu uvedenému ve schváleném harmonogramu projektu. Důvodem posunu termínu ukončení aktivity jsou zejména tyto skutečnosti:

- Rozsah realizace simulačních experimentů se ukázal časově náročnější, než bylo při návrhu projektu uvažováno.
- Na základě expertního posouzení výsledků projektu bylo řešitelským týmem přistoupeno k formulování a realizaci 3 sad dodatečných experimentů.

V rámci 4. monitorovacího období byly vyhodnocovány výsledky následujících experimentů:

- základní experimenty – zde patří experimenty, jejichž cílem bylo posouzení kvality stávajícího rozmístění posádek ZZS na území obou krajů. Dále byly realizovány experimenty vedoucí k návrhu geograficky optimálního rozmístění posádek ZZS na území obou krajů. Dále byly realizovány experimenty, v rámci kterých byla umožněna přeshraniční spolupráce v oblasti poskytování ZZS mezi oběma kraji a byl vyhodnocován dopad spolupráce na kvalitu ZZS v obou krajích. Všechny tyto návrhy byly poté ověřovány pomocí simulačních experimentů.
- dodatečné experimenty – tyto experimenty byly definovány v průběhu 4. monitorovacího období na základě připomínek z praxe a zejména jejich realizace si vyžádala posunutí termínu ukončení této aktivity. Výsledky optimalizačních experimentů ukázaly, že průměrný dojezdový čas je možno snížit až o cca 10 % v každém kraji. Toto snížení je však spíše teoretického charakteru, protože je spojeno s radikálními změnami v rozmístění výjezdových stanovišť ZZS (a jim přidělených posádek ZZS) provozovaných na území daného kraje, takže realizace takového počtu změn v systému ZZS je prakticky nerealizovatelná. Proto byly navrženy experimenty, které měly zejména ukázat vliv počtu změn v konfiguraci systému na snížení průměrného dojezdového času. Výsledky těchto dodatečných experimentů potom mohou sloužit jako podpůrný nástroj při rozhodování zřizovatelů a provozovatelů ZZS v případě, že by uvažovali o realizaci alespoň částečných změn v rozmístění posádek ZZS.

Výsledky všech provedených experimentů jsou obsahem souhrnné zprávy k výpočetní části projektu, která byla řešitelským týmem připravena. Tato zpráva je dostupná v plném znění na webových stránkách projektu.

Text programu InterregPart25

```
model 'InterregPart25_191125'
uses "mmxprs","mmive"; !gain access to the Xpress-Optimizer solver
```

```
! Preprogramovano dne 6.4.2025 podle getPartitioningQ z tx6vsk20/ISOLDE20/javafile
! Upraveno 060825 z InterregPart25upTo250425
! Preprogramovano 151125 z 'InterregPart25upTo060825'
! Preprogramovano 191125 z 'InterregPart_151125'
! Zmena spociva v pouziti vahy w misto poctu obyvatel b
procedure FullCircle(plot:integer, R:real, S1:real, S2:real)
  (!  Procedura vykresli plny kruh s polomerem R a se stredem
     v bode [S1, S2] . Barva je urcena prislusnou definici plot
  !)
  declarations
        Delt:real
        x:real
        Pi:real
  end-declarations
  Pi:=3.141592653589
  Delt:=Pi/90
  forall(k in 1..30)do
    x:=k*Delt
    IVEdrawline(plot,S1+R*cos(x),S2+R*sin(x), S1-R*cos(x), S2+R*sin(x))
    IVEdrawline(plot,S1+R*cos(x),S2-R*sin(x), S1-R*cos(x), S2-R*sin(x))
  end-do
end-procedure

procedure EmptyCircle(plot:integer, R:real, S1:real, S2:real)
  (!  Procedura vykresli kruznici s polomerem R a se stredem
     v bode [S1, S2] . Barva je urcena prislusnou definici plot
  !)
  declarations
        Delt:real
        x:real
        Pi:real
  end-declarations
  Pi:=3.141592653589
  Delt:=Pi/90
  forall(k in 1..45)do
    x:=k*Delt
    IVEdrawpoint(plot,S1+R*cos(x),S2+R*sin(x))
    IVEdrawpoint(plot,S1-R*cos(x), S2+R*sin(x))
    IVEdrawpoint(plot,S1+R*cos(x),S2-R*sin(x))
    IVEdrawpoint(plot,S1-R*cos(x), S2-R*sin(x))
  end-do
end-procedure

function EucDist(lng1:real,lat1:real,lng2:real,lat2:real):real
  (!  Funkce vyda vzdalenost v km dvou bodu zadanych souradnicemi
     [lng1, lat1] a [lng2, lat2], kde
    lat a lng jsou udany ve stupnich,  v desetinnem tvaru
  !)
  returned:= sqrt((lng1*111-lng2*111)^2+(lat1*111-lat2*111)^2)
end-function

procedure OrderIncReal(n:integer, U:array(r1:range)of integer, K:array(r2:range)of real)
```

```
  (! Tato procedura usporada realna cisla v K v bunkach 1, ..., n pomoci pole U, tak,
    aby platilo K(U(i))<= K(U(i+1)) pro i= 1..n-1
  !)
  declarations
        k:integer
        j:integer
  end-declarations
  forall(i in 1..n) U(i):=i
  if (n>1) then
    forall(i in 2..n) do
      k:=U(i)
      j:=i-1
      while ((j>0)and(K(U(j))>K(k))) do
        U(j+1):=U(j)
        j-=1
      end-do ! while j
      U(j+1):=k
    end-do ! forall i
  end-if ! if n>1
end-procedure

procedure OrderIncRealCol(n:integer, s: integer, U:array(r1:range)of integer,
                          K:array(r2:range, r3:range)of real)
  (! Tato procedura usporada realna cisla sloupci s matice K v bunkach 1, ..., n
    pomoci pole U, tak, aby platilo K(U(i),s)<= K(U(i+1),s) pro i= 1..n-1
  !)
  declarations
        k:integer
        j:integer
  end-declarations
  forall(i in 1..n) U(i):=i
  if (n>1) then
    forall(i in 2..n) do
      k:=U(i)
      j:=i-1
      while ((j>0)and(K(U(j),s)>K(k,s))) do
        U(j+1):=U(j)
        j-=1
      end-do ! while j
      U(j+1):=k
    end-do ! forall i
  end-if ! if n>1
end-procedure

procedure DrawPartitioning(n:integer,noU:integer,w:array(r1:range) of real,
              noJ:array(r4:range) of integer, sJ:array(r5:range,r6:range) of integer,
              lng:array(r2:range) of real,lat:array(r3:range) of real)
! Zobrazovani

  declarations
        maxlat:real ! maximalni zemepisna sirka komunit
        minlat:real ! minimalni zemepisna sirka komunit
```

```
        maxlng:real ! maximalni zemepisna delka komunit
        minlng:real ! minimalni zemepisna delka komunit
        plot1:integer
        plot2:integer
        plot3:integer
        plot4:integer
        plot5:integer
        plot6:integer
        plot7:integer
        plot8:integer
        plot9:integer
        R, S1, S2:real ! parametry kruhu (radius, zem. delka, zem. sirka
        maxw:real
  Pi:real ! Cislo pi
end-declarations

Pi:=3.141592653589



  ! Body [minlng,maxlat] a [maxlng,minlat] davaji postupne levy horni roh obrazku
  ! a pravy dolni roh obrazku.
  maxlat:= max(i in 1..n) lat(i)
  minlat:= min(i in 1..n) lat(i)
  maxlng:= max(i in 1..n) lng(i)
  minlng:= min(i in 1..n) lng(i)

  writeln
  writeln("maxlat ",maxlat)
  writeln("minlat ",minlat)
  writeln("maxlng ",maxlng)
  writeln("minlng ",minlng)

  plot1:=IVEaddplot("Input Data",IVE_WHITE)

  plot4:=IVEaddplot("Input Data",IVE_YELLOW)

  plot5:=IVEaddplot("Input Data",IVE_GREEN)
  plot6:=IVEaddplot("Input Data",IVE_RED)
  plot7:=IVEaddplot("Input Data",IVE_MAGENTA)
  plot8:=IVEaddplot("Input Data",IVE_BLUE)
  plot9:=IVEaddplot("Input Data",IVE_CYAN)

  ! Nakresleni bodu, ktere predstavuji levy horni a pravy dolni roh obrazku a
  ! urcuji jeho rozmer
  IVEdrawpoint(plot1,maxlng,minlat)
  IVEdrawpoint(plot1,minlng,maxlat)

  ! Computing of radii proportional to weights of communities
  ! Denotation of the weight of community by yellow circle
  maxw:= max(i in 1..n) w(i)
  forall(i in 1..n)do
```

```
      R:=0.005+(0.025*w(i))/maxw
      FullCircle(plot4, R, lng(i),lat(i))
    end-do

    ! Vykresleni casti
    forall(cast in 1..noU) do
      forall(i in 1.. noJ(cast)) do
        R:=0.007+(0.025*w(sJ(cast,i)))/maxw
        if(cast=1) then
          EmptyCircle(plot5, R,
            lng(sJ(cast,i)),lat(sJ(cast,i)))
        else
          if(cast=2) then
            EmptyCircle(plot6, R,
              lng(sJ(cast,i)),lat(sJ(cast,i)))
          else
            if(cast=3) then
              EmptyCircle(plot7, R,
                lng(sJ(cast,i)),lat(sJ(cast,i)))
            else
              if(cast=4) then
                EmptyCircle(plot8, R,
                  lng(sJ(cast,i)),lat(sJ(cast,i)))
              else
                    EmptyCircle(plot9, R,
                  lng(sJ(cast,i)),lat(sJ(cast,i)))
              end-if
            end-if
          end-if
        end-if
      end-do
    end-do

    ! Vykresleni komunit regionu cernymi body
    plot2:=IVEaddplot("Input Data",IVE_BLACK)
    forall(i in 1..n) IVEdrawpoint(plot2,lng(i),lat(i))

end-procedure

function mhod(A:integer,B:integer):integer
  if(A<B) then
    B:= A
  end-if
  returned:= B
end-function


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Konec definovani procedur a funkci
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

declarations
```

```
        PocKand:integer
        PocZak:integer  ! to same jako n
        p:integer ! pocet sanitek
        r:integer ! number of distiquished distance
        noExp:integer ! cislo experimentu
        noU:integer! pocet subregionu (casti)
        delta:real
        beta:real
        Sumb:integer ! Soucet obyvatel
        fp:integer ! celkovy fixovany pocet ambulanci
        lambda36:real
        mi36:real
        maxf:integer ! Maximal number of directly assigned ambulances
  maxA:real
  WW:string ! If "b" then number of inhabitants if "c" the number of trips
        JmnReg:string ! Jmeno regionu
        JmnRLP:string ! Jmeno souboru s RLP stanicemi
        JmnRZP:string ! Jmeno souboru s RZP stanicemi
        JmnRV:string ! Jmeno souboru s RV stanicemi, jen pro MSK
        JmnSucY:string ! Jmeno souboru se soucasnym stavem
        JmnPocObyv:string ! Jmeno souboru s pocty obyvatel
        JmnPocKNV:string ! Jmeno souboru s pocty vyjezdu
        JmnNazvyObci:string ! Jmeno souboru s nazvy obci
        JmnPartPart:string ! Jmeno souboru s parametry
        JmnGPS:string
        JmnDist:string
        ! Names of output files
        StrnoExp:string
        Jmn_noJ: string
  Jmn_sJ:string
  Jmn_qH:string
  Jmn_f: string
  Jmn_u:string
  Jmn_p:string
end-declarations

! Nacteni JmnReg a noExp z "ParametryExp.txt"
initializations from ("ParametryExp.txt")
        noExp
        JmnReg
        WW
end-initializations

! Definovani jmen vystupu
if(noExp<10) then StrnoExp:="00"+strfmt(noExp,1)
end-if
if(noExp>=10)and(noExp<100) then StrnoExp:="0"+strfmt(noExp,1)
end-if
if(noExp>=100)and(noExp<1000) then StrnoExp:=strfmt(noExp,1)
end-if

Jmn_noJ:= JmnReg+"_noJ_"+StrnoExp+WW+".txt"
```

```
Jmn_sJ:= JmnReg+"_sJ_"+StrnoExp+WW+".txt"
Jmn_qH:= JmnReg+"_qH_"+StrnoExp+WW+".txt"
Jmn_f:= JmnReg+"_f_"+StrnoExp+WW+".txt"
Jmn_u:= JmnReg+"_u_"+StrnoExp+WW+".txt"
Jmn_p:= JmnReg+"_p_"+StrnoExp+WW+".txt"
!JmnReg:="ZA"

!writeln(JmnReg, "  noExp", noExp)

JmnRLP:=JmnReg+"_PocetRLPObci.txt"
JmnRZP:=JmnReg+"_PocetRZPObci.txt"
JmnRV:=JmnReg+"_PocetRVObci.txt"
JmnSucY:=JmnReg+"_SoucasnyStavYref.txt"
JmnPocObyv:=JmnReg+"_PoctyObyv.txt"
JmnPocKNV:=JmnReg+"_KNVyjazdy.txt"
JmnNazvyObci:=JmnReg+"_NazvyObci.txt"
JmnParPart:=JmnReg+"_ParPart25_091125.txt"
JmnGPS:=JmnReg+"_GPSObci_blank.txt"
JmnDist:=JmnReg+"_Matica.txt"

!KONSTANTY A ROZMERY ULOHY
!dlzkaZony := 0.10
r:=5
! Podle udaju z 3.4.2025 od Ludky
! Intenzita prichodu pozadavek za minutu na jednu ambulanci v zilinskem regionu
lambda36:= 0.0010143611
! Intenzita vybavenych pozadavku za 1 minutu jednou ambulanci v regionu Zilina
! mi bylo vypocitano z P0= 0.798474
mi36:= 0.0040190396
!Pi:=3.141592653589
!Pi:=3.141592653589

delta:=0.01

fopen(JmnDist,F_INPUT)
   readln(PocKand)
   readln(PocZak)
fclose(F_INPUT)


writeln("PocKand ", PocKand, " PocZak ", PocZak)

declarations
        t:array(1..PocKand,1..PocZak) of real  ! matrix of time distances in minutes
        b:array(1..PocZak) of integer ! pocty obyvatel
        c:array(1..PocZak) of integer ! pocty volani
        w:array(1..PocZak) of integer !vahy komunit, bud b nebo c
        ! konstanty pro vykresleni
        lat:array(1..PocZak) of real ! Zemepisna sirka komunity v stupnich v desetinnem tvaru
        lng:array(1..PocZak) of real ! Zemepisna delka komunity v stupnich v desetinnem tvaru
        ! Vystupni pole
        f:array(1..PocKand) of integer !Lower bound of the number o ambulances
```

```
        u:array(1..PocKand) of integer !Upper bound of the number o ambulances
        ff:array(1..PocKand) of integer ! The current numbers of located ambulances
        RLP:array(1..PocKand) of integer! The number of RLP currently located
        RZP:array(1..PocKand) of integer! The number of RZP currently located
        RV:array(1..PocKand) of integer ! The number of RV currently located

        NodeName:array(1..PocZak) of string
end-declarations




! Reading of input structures from files
fopen(JmnDist,F_INPUT)
  readln
  readln
  forall(i in 1..PocKand,j in 1..PocZak) readln(t(i,j))
fclose(F_INPUT)

fopen(JmnPocObyv,F_INPUT)
  readln
  forall(i in 1..PocZak) readln(b(i))
fclose(F_INPUT)

fopen(JmnPocKNV,F_INPUT)
  readln
  forall(i in 1..PocZak) readln(c(i))
fclose(F_INPUT)

fopen(JmnGPS,F_INPUT)
  readln
  forall(i in 1..PocZak) readln(lat(i),lng(i))
fclose(F_INPUT)

fopen(JmnNazvyObci,F_INPUT)
  readln
  forall(i in 1..PocZak) readln(NodeName(i))
fclose(F_INPUT)

fopen(JmnRLP,F_INPUT)
  readln
  forall(i in 1..PocKand) readln(RLP(i))
fclose(F_INPUT)

fopen(JmnRZP,F_INPUT)
  readln
  forall(i in 1..PocKand) readln(RZP(i))
fclose(F_INPUT)
```

```
! Determination of p and current state of ambulance deployment ff
p:=0

if((JmnReg="MSK")or(JmnReg="MSKZA") )then
  fopen(JmnRV,F_INPUT)
    readln
    forall(i in 1..PocKand) readln(RV(i))
  fclose(F_INPUT)

  forall(i in 1..PocKand) do
    ff(i):=RLP(i)+RZP(i)+RV(i)
    p+=ff(i)
  end-do
else
  forall(i in 1..PocKand) do
    ff(i):=RLP(i)+RZP(i)
    p+=ff(i)
  end-do

end-if


writeln(" new p ", p)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  End of reading input data
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Decision of meaning of weights w (either b or c)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if(WW="b") then
  forall(i in 1..PocZak) w(i):=b(i)
else
        if (WW="c") then
          forall(i in 1..PocZak) w(i):=c(i)
        end-if
end-if

! Determination of minimal value od beta

Sumb:=0
forall(i in 1..PocZak) Sumb+=w(i)

fp:=0
forall(i in 1..PocZak) do
  fp+=floor(w(i)*p/Sumb)
end-do

! Percentage of  directively assigned ambulances
```

```
! Here, value of beta ensures the number of directly
! assigned ambulances without parameter of tolerance
beta:= ceil(100*fp/p)/100
! Here beta can be increased
beta+= 0.01

! Determination of the noU
 noU:=1+max(i in 1..PocZak) ceil(w(i)*p/Sumb)

! Kontrola parametru
writeln("p  ",p)
writeln("Sumb  ",Sumb)
writeln("Sumb/p  ",Sumb/p)
writeln("fp  ",fp)
writeln("beta  ",beta)



! Kontrola parametru

   writeln("PocKand  ",PocKand)
   writeln("PocZak  ",PocZak)
   writeln("p  ",p)
   writeln("r  ",r)
   writeln("noExp  ", noExp)
   writeln("noU  ", noU)
   writeln(" delta ", delta)
   writeln(" beta ", beta)



declarations
		noJ:array(1..noU) of integer ! pocet komunit v definovane casti reg.
		sJ:array(1..noU,1..PocZak)of integer ! seznam komunit v definovane casti reg.
		noJad:array(1..noU) of integer ! pocet jader (vybranych komunit) v casti regionu
		sJad:array(1..noU,1..PocZak)of integer ! seznam jader (komunit) v definovane casti reg.

		Zar: array(1..PocZak) of integer ! Pole signakizujici hodnotou 1, ze komunita
				! byla zarazena a hodnotou 0 opacny pripad

     ! Vystupni parametry
     TT: integer ! maximum z poctu elementu v jednotlivych castech dekompozice

     pp:integer
     ppp:integer

     q:array(1..noU,1..r) of real
end-declarations


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Rozklad mnoziny obci (verze z 080425)
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Inicializace poli f a u
forall(i in 1..PocKand) do
  f(i):=0
  u(i):=noU
end-do


! Determination of limit for exclusion community from negligible nodes,
! which can be excuded from ambulance location when a core is in  proximity of the node.



ppp:=floor(0.95*p)
!writeln(" r ",r, " pp ",pp, "  mhod(r,pp) ", mhod(r,pp))
alpha:=0.0
while(ppp>=sum(i in 1..PocZak) floor(alpha+w(i)*p/Sumb)) do
  writeln(" alpha ",alpha,"  sum  ",
           sum(i in 1..PocZak) floor(alpha+w(i)*p/Sumb) )
  alpha+=delta

end-do
if(alpha>0) then
  alpha-=delta
end-if
maxA:=alpha

writeln
writeln("maxA  ", maxA)
writeln

pp:=floor(beta*p)
!writeln(" r ",r, " pp ",pp, "  mhod(r,pp) ", mhod(r,pp))
alpha:=0.0
while(pp>=sum(i in 1..PocZak) floor(alpha+w(i)*p/Sumb)) do
  writeln(" alpha ",alpha,"  sum  ",
           sum(i in 1..PocZak) floor(alpha+w(i)*p/Sumb))
  alpha+=delta

end-do
if(alpha>0) then
  alpha-=delta
end-if

forall(i in 1..PocKand) f(i):=floor(alpha+w(i)*p/Sumb)
writeln
writeln(" alpha ",alpha)
forall(i in 1..PocKand) writeln(" i ",strfmt(i,4),
           " f(i) ",f(i)," ",NodeName(i))
writeln
writeln(" sum f(i)", sum(i in 1..PocKand) f(i))
```

```
maxf:=max(i in 1..PocKand) f(i)

writeln
writeln
writeln(" Sumb ",Sumb)
writeln(" Sumb/p ", Sumb/p)
writeln(" maxf ", maxf)

! Inicializace podle jiz zaradenych komunit
forall(i in 1..PocZak) Zar(i):=0

! Vlastni dekompozicni algoritmus produkujici casti regionu
! Urceni poctu casti
 !noU:=maxf+1

!Vytvoreni seznamu Jader
forall(cast in 1..maxf) noJad(cast):=0
forall(i in 1..PocKand|Zar(i)=0) do
  if(f(i)>0) then
     noJad(maxf+1-f(i))+=1
     sJad(maxf+1-f(i),noJad(maxf+1-f(i))) :=i
  end-if
end-do

! Definovani jednotlivych casti 1..noU-1
forall(cast in 1.. noU) noJ(cast):=0
forall(cast in 1.. noU-1|noJad(cast)>0) do
   ! Zpracovani seznamu jader
   forall(k in 1..noJad(cast)|Zar(sJad(cast,k))=0) do
      !Zasunuti jadra do Jcast
      noJ(cast)+=1
      sJ(cast,noJ(cast)):=sJad(cast,k)
      Zar(sJad(cast,k)):=1
      ! Zpracovani okoli jadra
      forall(i in 1..PocKand| (Zar(i)=0)and(f(i)=0)) do
         ! Zvetseni zakazaneho polomeru o 20% (to 1.2)
         if (t(sJad(cast,k),i)<=1.0*t(sJad(cast,k),sJad(cast,k))) then
            noJ(cast)+=1
            sJ(cast,noJ(cast)):=i
            Zar(i):=1
            ! Inclusion of i to the set of negligible communities
            if(floor(maxA+w(i)*p/Sumb)<1)then
               u(i):=0
            end-if
         end-if
      end-do
   end-do
end-do

! Definovani jednotlivych casti noU
forall(i in 1..PocKand|Zar(i)=0) do
   noJ(noU)+=1
```

```
   sJ(noU,noJ(noU)):=i
   Zar(i):=1
end-do




forall(cast in 1..noU| noJ(cast)>0) do
   writeln
   writeln
   forall(i in 1..noJ(cast)) writeln (" i ",strfmt(i,3),
               " sJ(cast,i) ", strfmt(sJ(cast,i),3),
          " b(sJ(cast,i)) ", strfmt( w(sJ(cast,i)),8),
                  " f(sJ(cast,i)) ", f(sJ(cast,i)),
                   " u(sJ(cast,i)) ",u(sJ(cast,i)))

end-do ! forall  cast

! Konec dekompozice


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Urceni  q(cast,*)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

 forall(cast in 1..noU) do
    q(cast,1):=mi36/(mi36+lambda36)
    forall(i in 2..r-1)q(cast,i):=q(cast,i-1)*(1-q(cast,1))
    q(cast,r):=1-sum(i in 1..r-1) q(cast,i)
 end-do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Konec urceni  q(cast, *)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Kontrola q

writeln
forall(cast in 1..noU) do
    forall(i in 1..r) writeln("q(",strfmt(cast,3),", ",strfmt(i,3)," ):= ",q(cast,i))

end-do


writeln
writeln(" pocet zaradenych ", sum(i in 1..PocZak) Zar(i))

! Kontrola rozkladu
writeln
writeln(" sum(i in 1.. noU) noJ(i) ",sum(i in 1.. noU) noJ(i))
```

```
! Vystupy do textovych souboru

! Output of the current state
fopen(JmnSucY,F_OUTPUT)
  writeln(PocKand)
  forall(i in 1..PocKand) writeln(ff(i))
fclose(F_OUTPUT)

fopen(Jmn_noJ,F_OUTPUT)
  writeln(noU)
  forall(i in 1..noU) writeln(noJ(i))
fclose(F_OUTPUT)

TT:=max(i in 1..noU) noJ(i)
fopen(Jmn_sJ,F_OUTPUT)
  writeln(noU)
  writeln(TT)
  forall(i in 1..noU) do
    forall(ii in 1..noJ(i)) writeln(sJ(i,ii))
    forall(ii in noJ(i)+1..TT ) writeln(0)
  end-do
fclose(F_OUTPUT)


fopen(Jmn_f,F_OUTPUT)
  writeln(PocKand)
  forall(i in 1..PocKand) writeln(f(i))
fclose(F_OUTPUT)


fopen(Jmn_u,F_OUTPUT)
  writeln(PocKand)
  forall(i in 1..PocKand) writeln(u(i))
fclose(F_OUTPUT)

fopen(Jmn_qH,F_OUTPUT)
  writeln(noU)
  writeln(r)
  forall(cast in 1..noU) do
    forall(i in 1..r) writeln(q(cast,i))
  end-do
fclose(F_OUTPUT)

fopen(Jmn_p,F_OUTPUT)
  writeln(1)
  writeln(p)
fclose(F_OUTPUT)


! Zobrazovani

!DrawPartitioning(n,noU,b,noJ, sJ,lng,lat)
```

```
! Zobrazovani

  declarations
      maxlat:real ! maximalni zemepisna sirka komunit
      minlat:real ! minimalni zemepisna sirka komunit
      maxlng:real ! maximalni zemepisna delka komunit
      minlng:real ! minimalni zemepisna delka komunit
      plot1:integer
      plot2:integer
      plot3:integer
      plot4:integer
      plot5:integer
      plot6:integer
      plot7:integer
      plot8:integer
      plot9:integer
      R, S1, S2:real ! parametry kruhu (radius, zem. delka, zem. sirka
      maxw:real
  Pi:real ! Cislo pi
end-declarations



  ! Body [minlng,maxlat] a [maxlng,minlat] davaji postupne levy horni roh obrazku
  ! a pravy dolni roh obrazku.
  maxlat:= max(i in 1..PocZak) lat(i)
  minlat:= min(i in 1..PocZak) lat(i)
  maxlng:= max(i in 1..PocZak) lng(i)
  minlng:= min(i in 1..PocZak) lng(i)

  writeln
  writeln("maxlat ",maxlat)
  writeln("minlat ",minlat)
  writeln("maxlng ",maxlng)
  writeln("minlng ",minlng)

  plot1:=IVEaddplot("Input Data",IVE_WHITE)

  plot4:=IVEaddplot("Input Data",IVE_YELLOW)

  plot5:=IVEaddplot("Input Data",IVE_GREEN)
  plot6:=IVEaddplot("Input Data",IVE_RED)
  plot7:=IVEaddplot("Input Data",IVE_MAGENTA)
  plot8:=IVEaddplot("Input Data",IVE_BLUE)
  plot9:=IVEaddplot("Input Data",IVE_CYAN)

  ! Nakresleni bodu, ktere predstavuji levy horni a pravy dolni roh obrazku a
  ! urcuji jeho rozmer
  IVEdrawpoint(plot1,maxlng,minlat)
  IVEdrawpoint(plot1,minlng,maxlat)
```

```
  ! Computing of radii proportional to weights of communities
  ! Denotation of the weight of community by yellow circle
  maxw:= max(i in 1..PocZak) w(i)
  forall(i in 1..PocKand)do
    R:=0.005+(0.025*w(i))/maxw
    FullCircle(plot4, R, lng(i),lat(i))
  end-do

  ! Vykresleni casti
  forall(cast in 1..noU) do
    forall(i in 1.. noJ(cast)) do
      R:=0.007+(0.025*w(sJ(cast,i)))/maxw
      if(cast=1) then
        EmptyCircle(plot5, R,
          lng(sJ(cast,i)),lat(sJ(cast,i)))
      else
        if(cast=2) then
          EmptyCircle(plot6, R,
            lng(sJ(cast,i)),lat(sJ(cast,i)))
        else
          if(cast=3) then
            EmptyCircle(plot7, R,
              lng(sJ(cast,i)),lat(sJ(cast,i)))
          else
            if(cast=4) then
              EmptyCircle(plot8, R,
                lng(sJ(cast,i)),lat(sJ(cast,i)))
            else
                  EmptyCircle(plot9, R,
                lng(sJ(cast,i)),lat(sJ(cast,i)))
            end-if
          end-if
        end-if
      end-if
    end-do
  end-do

  ! Vykresleni komunit regionu cernymi body
  plot2:=IVEaddplot("Input Data",IVE_BLACK)
  forall(i in 1..PocZak) IVEdrawpoint(plot2,lng(i),lat(i))


end-model !'InterregPart25_191125'
```

Text programu InterregOpt25

```
model "InterregOpt25_191125"
! Prepared for FailingCenters23 on March 2, 2023
! The programm was created by reprogramming "InterregOpt25_Rad_Loc" on November 15, 2025
!
! Preprogramovano 191125 z 'InterregOpt25_151125


uses "mmxprs", "mmsystem", "mmive";

! FUNKCIE

!*******************************************************************************************

!*******************************************************************************************

!*******************************************************************************************

declarations
  optUloha3: mpproblem!Optimalizacna uloha (pre potreby procedur)
  optUloha4: mpproblem!Optimalizacna uloha (pre potreby procedur)
end-declarations



! The function FNCGPH computes the value of generalized disutility for
! stochastically heterogeneous region of a weighted
! p-median solution determined by two lists sY and sF, where p elements of
! sY subscripted from 1 to p are row numbers of the matrix dij and corresponds to
! service center locations. The list sF contains numbers of facilities located
! at the centers.
! The function returns also iY, which is vector of all possible center locations
! with the associated numbers of facilities. Further, array minr (r+1xn) is returned.
! j-th column of the matrix contains r+1 subscripts of center locations (rows of matrix dij)
! from sY, which belongs to r+1 centers nearest to user location j. The subscripts are ordered
! according to increasing distance from location j.
! Repaired 240325

function FNCGPH(
        I: range, J: range, R: range, p: integer, r: integer,noU:integer,
        q: array(r1: range, r13: range) of real,
        dij: array(r2: range, r3: range) of real,
        noJ:array(r10: range) of integer, ! noJ(u) the number of elements of the part u
        sJ:array(r11: range, r12: range) of integer, ! sJ(u) the list of locations of the part u
        obyv: array(r4: range) of integer,
        noY:integer,sY:array(r5:range) of integer,
        sF:array(r6:range) of integer,iY:array(r7:range) of integer,
        minr:array(r8:range,r9:range) of integer): real

        declarations
                vysl:real
                nom:integer
                ic:integer
```

```
            k:integer
            j:integer
            last:integer
            ss:real
            cc:integer
        end-declarations

        ! Transformation of sY and sF to iY
        forall(i in I) iY(i):=0
        forall(i in 1..noY) iY(sY(i)):=sF(i)

vysl := 0.0
forall(u1 in 1..noU)
 forall(ii in 1.. noJ(u1)) do ! processing of j th column
  j:=sJ(u1,ii)
  nom :=1 ! the number of currently inserted centers
  minr(1,j):=sY(1) ! initialization

  forall(i in 2..noY ) do! ordering of the r+1 closest locations
    ic:=sY(i) ! ith center is being inserted or refused
    dic:=dij(ic,j)
    ! Search for the first k, where inserted center should be placed
    k:=1
    while((k<=nom)and(dic>dij(minr(k,j),j))) k+=1


    if(k>nom) then ! the  currently inserted nom centers are closer to j then ic
      if (nom<=r) then ! ic is inserted into list of r+1 closest centers
        nom+=1      ! as the nom+1 th element
        minr(nom,j):=ic

      end-if


    else ! k<=nom the inspected center ic should be inserted at k th position
       ! the inserted centers must be shifted by one from the position k
       ! to the position nom if nom<=r, if nom=r+1, then they are shifted
       ! from k to r (the r+1 element is rewriten). The list has at most r+1
       ! members.
       if(nom<=r) then
         last:=nom
         nom+=1
       else ! if nom>r i.e. nom=r+1, the list is full
              last:=r
       end-if
       forall(kk in 0..last-k) minr(last+1-kk,j):=minr(last-kk,j)
       minr(k,j):=ic ! insertion of ic at the released position


    end-if ! k=nom else
  end-do ! forall i
```

```
      ! Here, there is r+1 centers from sY closest to j in the list minr(*,j) so
      ! that the nearest one is at the position 1 and the farthest one at the
      ! position r+1

      ! The approach, which takes into account the multiple facility assignment
      ss:=0
      k:=1 ! subscript of processed facility
      last:=0 ! an upper bound of k for processed center minr(cc)
      cc:=1 !subscript of processed center
      while(k<=r) do
        last+=iY(minr(cc,j)) ! Repaired 240325 cc for k
        if(last>r) then last:=r
        end-if
        while(k<=last) do
          ss+= dij(minr(cc,j),j)*q(u1,k)
          k+=1
        end-do
        cc+=1
      end-do ! while k
      vysl+=obyv(j)*ss
    end-do ! forall j
    returned := round(vysl)
end-function ! end of FNCGPH


! The function FNCGPH computes the average response time value of generalized disutility for
! stochastically heterogeneous region of a weighted
! p-median solution determined by two lists sY and sF, where p elements of
! sY subscripted from 1 to p are row numbers of the matrix dij and corresponds to
! service center locations. The list sF contains numbers of facilities located
! at the centers.
! The function returns also iY, which is vector of all possible center locations
! with the associated numbers of facilities. Further, array minr (r+1xn) is returned.
! j-th column of the matrix contains r+1 subscripts of center locations (rows of matrix dij)
! from sY, which belongs to r+1 centers nearest to user location j. The subscripts are ordered
! according to increasing distance from location j.
! Repaired 240325 adjusted 060825

function FNCGPHavg(
          I: range, J: range, R: range, p: integer, r: integer,noU:integer,
          q: array(r1: range, r13: range) of real,
          dij: array(r2: range, r3: range) of real,
          noJ:array(r10: range) of integer, ! noJ(u) the number of elements of the part u
          sJ:array(r11: range, r12: range) of integer, ! sJ(u) the list of locations of the part u
          w: array(r4: range) of integer,
          noY:integer,sY:array(r5:range) of integer,
          sF:array(r6:range) of integer,iY:array(r7:range) of integer,
          minr:array(r8:range,r9:range) of integer): real

          declarations
```

```
                vysl:real
                nom:integer
                ic:integer
                k:integer
                j:integer
                last:integer
                ss:real
                cc:integer
                W:real
        end-declarations

        W:=0
        forall(j1 in J) W+=w(j1)

        ! Transformation of sY and sF to iY
        forall(i in I) iY(i):=0
        forall(i in 1..noY) iY(sY(i)):=sF(i)

vysl := 0.0
forall(u1 in 1..noU)
 forall(ii in 1.. noJ(u1)) do ! processing of j th column
  j:=sJ(u1,ii)
  nom :=1 ! the number of currently inserted centers
  minr(1,j):=sY(1) ! initialization

  forall(i in 2..noY ) do! ordering of the r+1 closest locations
    ic:=sY(i) ! ith center is being inserted or refused
    dic:=dij(ic,j)
    ! Search for the first k, where inserted center should be placed
    k:=1
    while((k<=nom)and(dic>dij(minr(k,j),j))) k+=1


    if(k>nom) then ! the  currently inserted nom centers are closer to j then ic
      if (nom<=r) then ! ic is inserted into list of r+1 closest centers
        nom+=1      ! as the nom+1 th element
        minr(nom,j):=ic

      end-if


    else ! k<=nom the inspected center ic should be inserted at k th position
       ! the inserted centers must be shifted by one from the position k
       ! to the position nom if nom<=r, if nom=r+1, then they are shifted
       ! from k to r (the r+1 element is rewriten). The list has at most r+1
       ! members.
       if(nom<=r) then
         last:=nom
         nom+=1
       else ! if nom>r i.e. nom=r+1, the list is full
                last:=r
       end-if
```

```
          forall(kk in 0..last-k) minr(last+1-kk,j):=minr(last-kk,j)
          minr(k,j):=ic ! insertion of ic at the released position


        end-if ! k=nom else
      end-do ! forall i



      ! Here, there is r+1 centers from sY closest to j in the list minr(*,j) so
      ! that the nearest one is at the position 1 and the farthest one at the
      ! position r+1

      ! The approach, which takes into account the multiple facility assignment
      ss:=0
      k:=1 ! subscript of processed facility
      last:=0 ! an upper bound of k for processed center minr(cc)
      cc:=1 !subscript of processed center
      while(k<=r) do
        last+=iY(minr(cc,j)) ! Repaired 240325 cc for k
        if(last>r) then last:=r
        end-if
        while(k<=last) do
          ss+= dij(minr(cc,j),j)*q(u1,k)
          k+=1
        end-do
        cc+=1
      end-do ! while k
      vysl+=w(j)*ss
    end-do ! forall j
    returned := vysl/W
end-function ! end of FNCGPHavg

!*****************************************************************************************

! The function computes location problem using the radial approach.
! The length of gap between radii is optional (LZ).
! The function was completed on August 5, 2025
! Resulting value is sum of the expected response times

function getWPMGPHrad25(I: range, J: range, r: integer, p: integer, !Zakladne rozmery ulohy
          m: integer, !maximal considered distance
          N:integer, B:integer,D:integer, ! NEW  Strength of fainess, B max no out of radius D
          noU:integer, ! The number of parts in the partitioning
          TT:integer,        ! Maximal number of locations of a part
          q: array(r1: range,r7: range) of real,  !Redukcne koeficienty pre generalized disutility
          b: array(r2: range) of integer,        !Pocty obyvatelov
          d: array(r3: range, r4: range) of real, !Matica vzdialenosti (casov)
          noJ:array(r8: range) of integer, ! noJ(u) the number of elements of the part u
          sJ:array(r9: range, r10: range) of integer, ! sJ(u) the list of locations of the part u
          f:array(r11:range) of integer, ! NEW minimal number of located facilities
          u:array(r12:range) of integer, ! NEW maximalnumber of located facilities
```

```
    LZ:real, ! Length of Zone i.e. distance between succeeding radii
    sI1: array(r5: range) of integer,      !Vystupny zoznam vybudovanych centier
    sF1: array(r6: range) of integer):real  !Vystupny zoznam poctov vozidiel v centrach
    with optUloha3 do

declarations

  navratovaHodnota: real !Navratova hodnota (ucelova funkcia alebo -1)

        R = 1..r
        M = 0..m
        T = 0..m-1
        A: array(M, I, J) of integer !Matica pokrytia - del. body
        pom: integer             !Pomocna premenna

        !Premenne modelu
        x: array(J, T, R) of mpvar
        y: array(I) of mpvar
        ! NEW after March 2, 2023
        z: array(J) of mpvar
      end-declarations



    !Vypocet matice pokrytia
forall(t in M, i in I, j in J)
  if d(i, j) <= (t * LZ)
  then A(t, i, j) := 1
  else A(t, i, j) := 0
end-if

      !MATEMATICKY MODEL ULOHY


      !Obligatory constraints
    forall(j in J, t in T, k in R)
        RngX(j, t, k) := x(j, t, k) is_binary

      forall(i in I)
        RngY(i) := y(i) is_integer  ! The  change

      forall(j in J)
        RngZ(j) := z(j) is_binary  ! The  change after March 2, 2023


      ! Objective function H for stochastically heterogeneous region
      ObjF := sum(u1 in 1..noU, j in 1..noJ(u1), k in R, t in T)
                        (b(sJ(u1,j)) * q(u1,k)* x(sJ(u1,j), t, k) * LZ)

      !Structural constraints
      forall(j in J, t in T)
  Cov(j, t) := sum(k in R) x(j, t, k) + sum(i in I) A(t, i, j)*y(i) >= r
```

```
NoF := sum(i in I) y(i) = p

! NEW after March 2, 2023

forall(j in J)
  CovZ(j) := N*z(j)+ sum(i in I) A(D, i, j)*y(i) >= N

FairB:= sum(j in J) b(j)*z(j) <= B

forall(i in I)
          LLimY(i) := y(i) >= f(i)
        forall(i in I)
          ULimY(i) := y(i) <= u(i)


!Optimalizacia
        !startTime := gettime

minimize(ObjF)

!gloCompTime := gettime - startTime

        if (getprobstat = XPRS_INF)
        then navratovaHodnota := -1
        else navratovaHodnota := getobjval

          !Zoznam vybudovanych stredisk
          forall(i in 1..p) do
            sI1(i):=0
            sF1(i):=0
          end-do
          pom := 0
          forall(i in I|getsol(y(i)) > 0.1) do
    pom += 1
    sI1(pom) := i
    sF1(pom) := round(getsol(y(i)))
  end-do
        end-if

        !Cleaning
        !Objective function
        ObjF := 0

        !Structural constraints
        forall(j in J, t in T)
  Cov(j, t) := 0

        NoF := 0

        forall(j in J)
  CovZ(j) := 0
```

```
    FairB:= 0

    forall(i in I)
            LLimY(i) := 0
          forall(i in I)
            ULimY(i) := 0

        !Obligatory constraints
          forall(j in J, t in T, k in R)
            RngX(j, t, k) := 0

          forall(i in I) RngY(i) := 0

          forall(j in J)
            RngZ(j) := 0



  end-do ! with OptUloha3
  returned := navratovaHodnota

end-function ! end of getWPMGPHrad23




! The function computes location problem using the location-allocation approach.
! The length of gap between radii is optional (LZ).
! The function was completed on August 5, 2025
! Resulting value is sum of the expected response times

function getWPMGPHloc25(I: range, J: range, r: integer, p: integer, !Zakladne rozmery ulohy
        m: integer, !maximal considered distance
        N:integer, B:integer,D:integer, ! NEW  Strength of fainess, B max no out of radius D
        noU:integer, ! The number of parts in the partitioning
        TT:integer,        ! Maximal number of locations of a part
        q: array(r1: range,r7: range) of real,  !Redukcne koeficienty pre generalized disutility
        b: array(r2: range) of integer,        !Pocty obyvatelov
        d: array(r3: range, r4: range) of real, !Matica vzdialenosti (casov)
        noJ:array(r8: range) of integer, ! noJ(u) the number of elements of the part u
        sJ:array(r9: range, r10: range) of integer, ! sJ(u) the list of locations of the part u
        f:array(r11:range) of integer, ! NEW minimal number of located facilities
        u:array(r12:range) of integer, ! NEW maximalnumber of located facilities
        sI1: array(r5: range) of integer,      !Vystupny zoznam vybudovanych centier
        sF1: array(r6: range) of integer):real  !Vystupny zoznam poctov vozidiel v centrach
        with optUloha4 do

    declarations
      navratovaHodnota: real !Navratova hodnota (ucelova funkcia alebo -1)

            R = 1..r
```

```
            pom: integer            !Pomocna premenna

            !Premenne modelu
            x: array(J) of mpvar ! if x=1 then community j is far than D from the assigned
ambulance
            y: array(I) of mpvar
            ! NEW after August 4, 2025
            z: array(I,J,R) of mpvar
          end-declarations

          !MATEMATICKY MODEL ULOHY

          !Obligatory constraints
        forall(i in I,j in J, k in R)
            RngZ(i, j, k) := z(i, j, k) is_binary

          forall(i in I)
            RngY(i) := y(i) is_integer

          !Objective function
          ObjF := sum(u1 in 1..noU,j in  1..noJ(u1), k in R, i in I) (b(sJ(u1,j)) * q(u1,k)*d(i,sJ(u1,j))*
z(i,sJ(u1,j),k))

          !Structural constraints


    NoF := sum(i in I) y(i) = p

    forall(j in J, k in R)
      Ass(j, k) := sum(i in I) z(i,j,k) = 1

    forall(j in J, i in I)
      Link(j, i) := sum(k in R) z(i,j,k) <= y(i)

     forall(j in J)
      CovX(j) := sum(i in I) d(i,j)*z(i,j,1)<=D+m*x(j)

    FairB:= sum(j in J) b(j)*x(j) <= B

    ! Limitation of the number of located ambulances
          forall(i in I)
            LLimY(i) := y(i) >= f(i)
          forall(i in I)
            ULimY(i) := y(i) <= u(i)

          forall(j in J)
            RngX(j) := x(j) is_binary  ! The  change after March 2, 2023


    !Optimalizacia
```

```
minimize(ObjF)



        if (getprobstat = XPRS_INF)
        then navratovaHodnota := -1
        else navratovaHodnota := getobjval

          !Zoznam vybudovanych stredisk
          forall(i in 1..p) do
            sI1(i):=0
            sF1(i):=0
          end-do
          pom := 0
          forall(i in I|getsol(y(i)) > 0.1) do
      pom += 1
      sI1(pom) := i
      sF1(pom) := round(getsol(y(i)))
    end-do
        end-if

        !Cleaning

        ObjF := 0

        !Structural constraints
        NoF := 0

        forall(j in J)
    CovX(j) := 0

FairB:= 0

forall(i in I)
          LLimY(i) := 0
        forall(i in I)
          ULimY(i) := 0
        forall(j in J, k in R)
    Ass(j, k) :=0

forall(j in J, i in I)
    Link(j, i) :=0

      !Obligatory constraints

        forall(j in J)
          RngX(j) := 0

        forall(i in I) RngY(i) := 0

        forall(i in I,j in J, k in R)
          RngZ(i, j, k) := 0
```

```
  end-do ! with OptUloha4
  returned := navratovaHodnota

end-function ! end of getWPMGPHloc25

! The function computes location problem using the radial approach.
! The length of gap between radii is optional (LZ).
! The function was completed on August 5, 2025
! Resulting value is average expected response time

function getWPMGPHrad25avg(I: range, J: range, r: integer, p: integer, !Zakladne rozmery ulohy
        m: integer, !maximal considered distance
        N:integer, B:integer,D:integer, ! NEW  Strength of fainess, B max no out of radius D
        noU:integer, ! The number of parts in the partitioning
        TT:integer,        ! Maximal number of locations of a part
        q: array(r1: range,r7: range) of real,  !Redukcne koeficienty pre generalized disutility
        w: array(r13: range) of integer, ! weight of individual community
        b: array(r2: range) of integer,       !Pocty obyvatelov
        d: array(r3: range, r4: range) of real, !Matica vzdialenosti (casov)
        noJ:array(r8: range) of integer, ! noJ(u) the number of elements of the part u
        sJ:array(r9: range, r10: range) of integer, ! sJ(u) the list of locations of the part u
        f:array(r11:range) of integer, ! NEW minimal number of located facilities
        u:array(r12:range) of integer, ! NEW maximalnumber of located facilities
        LZ:real, ! Length of Zone i.e. distance between succeeding radii
        sI1: array(r5: range) of integer,      !Vystupny zoznam vybudovanych centier
        sF1: array(r6: range) of integer):real  !Vystupny zoznam poctov vozidiel v centrach
        with optUloha3 do

    declarations

      navratovaHodnota: real !Navratova hodnota (ucelova funkcia alebo -1)

          R = 1..r
          M = 0..m
          T = 0..m-1
          A: array(M, I, J) of integer !Matica pokrytia - del. body
          pom: integer             !Pomocna premenna

          !Premenne modelu
          x: array(J, T, R) of mpvar
          y: array(I) of mpvar
          ! NEW after March 2, 2023
          z: array(J) of mpvar
          W:real ! Sum of the weights
        end-declarations

      W:= 0
      forall(j in J) W+=w(j)
```

```
    !Vypocet matice pokrytia
forall(t in M, i in I, j in J)
  if d(i, j) <= (t * LZ)
  then A(t, i, j) := 1
  else A(t, i, j) := 0
end-if

      !MATEMATICKY MODEL ULOHY


      !Obligatory constraints
    forall(j in J, t in T, k in R)
        RngX(j, t, k) := x(j, t, k) is_binary

      forall(i in I)
        RngY(i) := y(i) is_integer  ! The  change

      forall(j in J)
        RngZ(j) := z(j) is_binary  ! The  change after March 2, 2023


      ! Objective function H for stochastically heterogeneous region
      ObjF := sum(u1 in 1..noU, j in 1..noJ(u1), k in R, t in T)
            (w(sJ(u1,j)) * q(u1,k)* x(sJ(u1,j), t, k) * LZ)

      !Structural constraints
      forall(j in J, t in T)
  Cov(j, t) := sum(k in R) x(j, t, k) + sum(i in I) A(t, i, j)*y(i) >= r

NoF := sum(i in I) y(i) = p

! NEW after March 2, 2023

forall(j in J)
  CovZ(j) := N*z(j)+ sum(i in I) A(D, i, j)*y(i) >= N

FairB:= sum(j in J) b(j)*z(j) <= B

forall(i in I)
        LLimY(i) := y(i) >= f(i)
      forall(i in I)
        ULimY(i) := y(i) <= u(i)


!Optimalizacia
      !startTime := gettime

minimize(ObjF)

!gloCompTime := gettime - startTime

      if (getprobstat = XPRS_INF)
```

```
            then navratovaHodnota := -1
            else navratovaHodnota := getobjval/W

              !Zoznam vybudovanych stredisk
              forall(i in 1..p) do
                sI1(i):=0
                sF1(i):=0
              end-do
              pom := 0
              forall(i in I|getsol(y(i)) > 0.1) do
          pom += 1
          sI1(pom) := i
          sF1(pom) := round(getsol(y(i)))
        end-do
            end-if

            !Cleaning
            !Objective function
            ObjF := 0

            !Structural constraints
            forall(j in J, t in T)
        Cov(j, t) := 0

            NoF := 0

            forall(j in J)
        CovZ(j) := 0

      FairB:= 0

      forall(i in I)
              LLimY(i) := 0
            forall(i in I)
              ULimY(i) := 0

          !Obligatory constraints
            forall(j in J, t in T, k in R)
              RngX(j, t, k) := 0

            forall(i in I) RngY(i) := 0

            forall(j in J)
              RngZ(j) := 0



    end-do ! with OptUloha3
    returned := navratovaHodnota

end-function ! end of getWPMGPHrad25avg
```

```
! The function computes location problem using the location-allocation approach.
! The length of gap between radii is optional (LZ).
! The function was completed on August 5, 2025
! Resulting value is average expected response time

function getWPMGPHloc25avg(I: range, J: range, r: integer, p: integer, !Zakladne rozmery ulohy
        m: integer, !maximal considered distance
        N:integer, B:integer,D:integer, ! NEW  Strength of fainess, B max no out of radius D
        noU:integer, ! The number of parts in the partitioning
        TT:integer,        ! Maximal number of locations of a part
        q: array(r1: range,r7: range) of real,  !Redukcne koeficienty pre generalized disutility
        w: array(r13: range) of integer, ! weight of individual community
        b: array(r2: range) of integer,        !Pocty obyvatelov
        d: array(r3: range, r4: range) of real, !Matica vzdialenosti (casov)
        noJ:array(r8: range) of integer, ! noJ(u) the number of elements of the part u
        sJ:array(r9: range, r10: range) of integer, ! sJ(u) the list of locations of the part u
        f:array(r11:range) of integer, ! NEW minimal number of located facilities
        u:array(r12:range) of integer, ! NEW maximalnumber of located facilities
        sI1: array(r5: range) of integer,      !Vystupny zoznam vybudovanych centier
        sF1: array(r6: range) of integer):real  !Vystupny zoznam poctov vozidiel v centrach
        with optUloha4 do

    declarations
      navratovaHodnota: real !Navratova hodnota (ucelova funkcia alebo -1)

            R = 1..r

            pom: integer            !Pomocna premenna

            !Premenne modelu
            x: array(J) of mpvar ! if x=1 then community j is far than D from the assigned
ambulance
            y: array(I) of mpvar
            ! NEW after August 4, 2025
            z: array(I,J,R) of mpvar
             W:real ! Sum of the weights
    end-declarations
    W:= 0
    forall(j in J) W+=w(j)

          !MATEMATICKY MODEL ULOHY

          !Obligatory constraints
        forall(i in I,j in J, k in R)
            RngZ(i, j, k) := z(i, j, k) is_binary

          forall(i in I)
            RngY(i) := y(i) is_integer
```

```
        !Objective function
        ObjF := sum(u1 in 1..noU,j in  1..noJ(u1), k in R, i in I) (w(sJ(u1,j)) * q(u1,k)*d(i,sJ(u1,j))*
z(i,sJ(u1,j),k))

        !Structural constraints


    NoF := sum(i in I) y(i) = p

    forall(j in J, k in R)
      Ass(j, k) := sum(i in I) z(i,j,k) = 1

    forall(j in J, i in I)
      Link(j, i) := sum(k in R) z(i,j,k) <= y(i)

     forall(j in J)
      CovX(j) := sum(i in I) d(i,j)*z(i,j,1)<=D+m*x(j)

    FairB:= sum(j in J) b(j)*x(j) <= B

    ! Limitation of the number of located ambulances
        forall(i in I)
          LLimY(i) := y(i) >= f(i)
        forall(i in I)
          ULimY(i) := y(i) <= u(i)

        forall(j in J)
          RngX(j) := x(j) is_binary  ! The  change after March 2, 2023


    !Optimalizacia

    minimize(ObjF)



        if (getprobstat = XPRS_INF)
        then navratovaHodnota := -1
        else navratovaHodnota := getobjval/W

          !Zoznam vybudovanych stredisk
          forall(i in 1..p) do
            sI1(i):=0
            sF1(i):=0
          end-do
          pom := 0
          forall(i in I|getsol(y(i)) > 0.1) do
      pom += 1
      sI1(pom) := i
      sF1(pom) := round(getsol(y(i)))
    end-do
        end-if
```

```
		!Cleaning

		ObjF := 0

		!Structural constraints
		NoF := 0

		forall(j in J)
	   CovX(j) := 0

	FairB:= 0

	forall(i in I)
		   LLimY(i) := 0
		forall(i in I)
		   ULimY(i) := 0
		forall(j in J, k in R)
	   Ass(j, k) :=0

	forall(j in J, i in I)
	   Link(j, i) :=0

	      !Obligatory constraints

		forall(j in J)
		   RngX(j) := 0

		forall(i in I) RngY(i) := 0

		forall(i in I,j in J, k in R)
		   RngZ(i, j, k) := 0


   end-do ! with OptUloha4
   returned := navratovaHodnota

end-function ! end of getWPMGPHloc25avg


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Konec definovani procedur a funkci
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!



declarations
	PocKand:integer
	PocZak:integer  ! to same jako n
	p:integer ! pocet sanitek
	r:integer ! number of distiquished distance
	noExp:integer ! cislo experimentu
```

```
        noU:integer! pocet subregionu (casti)
        sumB:real ! Soucet obyvatel
        fp:integer ! fixovany pocet ambulanci
        TT: integer !Maximal number of locations in a part ziskam z sJ, jako druhy radek
        WW:string ! If "b" then number of inhabitants if "c" the number of trips
        JmnReg:string ! Jmeno regionu
        JmnRLP:string ! Jmeno souboru s RLP stanicemi
        JmnRZP:string ! Jmeno souboru s RZP stanicemi
        JmnRV:string ! Jmeno souboru s RV stanicemi, jen pro MSK
        JmnSucY:string ! Jmeno souboru se soucasnym stavem
        JmnPocObyv:string ! Jmeno souboru s pocty obyvatel
        JmnPocKNV:string ! Jmeno souboru s pocty vyjezdu
        JmnNazvyObci:string ! Jmeno souboru s nazvy obci
        JmnPartPart:string ! Jmeno souboru s parametry
        JmnGPS:string
  JmnDist:string
  ! Names of input files
        Jmn_noJ: string
  Jmn_sJ:string
  Jmn_qH:string
  Jmn_f: string
  Jmn_u:string
  Jmn_p:string

end-declarations

! Nacteni JmnReg a noExp z "ParametryExp.txt"
initializations from ("ParametryExp.txt")
        noExp
        JmnReg
        WW
end-initializations

!JmnReg:="ZA"

writeln(JmnReg, "  noExp  ", noExp)

! Definovani jmen vstupu
if(noExp<10) then StrnoExp:="00"+strfmt(noExp,1)
end-if
if(noExp>=10)and(noExp<100) then StrnoExp:="0"+strfmt(noExp,1)
end-if
if(noExp>=100)and(noExp<1000) then StrnoExp:=strfmt(noExp,1)
end-if

Jmn_noJ:= JmnReg+"_noJ_"+StrnoExp+WW+".txt"
Jmn_sJ:= JmnReg+"_sJ_"+StrnoExp+WW+".txt"
Jmn_qH:= JmnReg+"_qH_"+StrnoExp+WW+".txt"
Jmn_f:= JmnReg+"_f_"+StrnoExp+WW+".txt"
Jmn_u:= JmnReg+"_u_"+StrnoExp+WW+".txt"
Jmn_p:= JmnReg+"_p_"+StrnoExp+WW+".txt"
```

```
! Definovani jmen globalnich vstupu
JmnRLP:=JmnReg+"_PocetRLPObci.txt"
JmnRZP:=JmnReg+"_PocetRZPObci.txt"
JmnRV:=JmnReg+"_PocetRVObci.txt"
JmnSucY:=JmnReg+"_SoucasnyStavYref.txt"
JmnPocObyv:=JmnReg+"_PoctyObyv.txt"
JmnPocKNV:=JmnReg+"_KNVyjazdy.txt"
JmnNazvyObci:=JmnReg+"_NazvyObci.txt"
JmnParPart:=JmnReg+"_ParPart25_091125.txt"
JmnGPS:=JmnReg+"_GPSObci_blank.txt"
JmnDist:=JmnReg+"_Matica.txt"



!Nazvy vstupnych a vystupnych suborov

JmnSolution :=JmnReg+"_"+StrnoExp+WW+ 'ApproxSolution.txt'    !Vystupny subor pre najdene
riesenie
JmnYref :=JmnReg+"_"+StrnoExp+WW+ 'ApproxSolYref.txt'        !Vystupny subor pre najdene
riesenie (len hodnoty premennych y)
JmnSummary := JmnReg+"_"+StrnoExp+WW+'ApproxSummary.txt'     !Sumarny vystup (pre
vsetky parametre)
!JmnSucY := JmnReg+"_SucasnyStavYref.txt"
JmnVystupProtokol := JmnReg+"_"+StrnoExp+WW+"_protokol"
JmnOut:= JmnReg+"_"+StrnoExp+WW+"_Output.txt"
!-------------------------------------------------------------------------------

!KONSTANTY
dlzkaZony := 0.10
r:=5

!PARAMETRE ULOHY

! Zjisteni PocKand a PocZak
fopen(JmnDist,F_INPUT)
   readln(PocKand)
   readln(PocZak)
fclose(F_INPUT)

! Zjisteni TT a noU
fopen(Jmn_sJ,F_INPUT)
   readln(noU)
   readln(TT)
fclose(F_INPUT)

! Zjisteni p

fopen(Jmn_p,F_INPUT)
   readln
   readln(p)
fclose(F_INPUT)
```

```
declarations
 startTime: real
 compTime: real

 ! added on March 2, 2023
 N:integer
 B:integer
 D:integer
 n:integer !Pocet zakazniku

end-declarations



! Zjisteni TT a noU
fopen(Jmn_sJ,F_INPUT)
   readln(noU)
   readln(TT)
fclose(F_INPUT)

! Zjisteni p

fopen(Jmn_p,F_INPUT)
   readln
   readln(p)
fclose(F_INPUT)

! Kontrola parametru

   writeln("PocKand  ",PocKand)
   writeln("PocZak  ",PocZak)
   writeln("p  ",p)
   writeln("r  ",r)
   writeln("noExp  ", noExp)
   writeln("noU  ", noU)

n:=PocZak

declarations
 I0 = 1..PocKand
 J0 = 1..PocZak
 !Konstanty
 b: array(J0) of integer ! Pocty obyvatel
 c: array(J0) of integer ! Pocty KNV vyjezdu
 w: array(J0) of integer
 sumbj: integer

 dij:array(I0, J0) of real
 refY:array(I0) of integer
 noJ:array(1..noU) of integer
```

```
  sJ:array(1..noU,1..TT) of integer
  q:array(1..noU,1..r) of real
        !added on March 2, 2023
  ff: array(I0) of integer ! real numbers of facilities
  f: array(I0) of integer ! lower limit of facilities
  u: array(I0) of integer ! upper limit of facilities
  NodeName: array(J0) of string

 ! konstanty pro vykresleni
        lat:array(1..n) of real ! Zemepisna sirka komunity v stupnich v desetinnem tvaru
        lng:array(1..n) of real ! Zemepisna delka komunity v stupnich v desetinnem tvaru
end-declarations

! Cislo Pi
!Pi:=3.141592653589

!Nacitanie konkretnej matice
fopen(JmnDist, F_INPUT)
  readln
  readln
  forall(i in I0, j in J0)readln(dij(i, j))
fclose(F_INPUT)


fopen(JmnPocObyv, F_INPUT)
  readln !Pocet zakaznikov
  forall(j in J0) readln(b(j))
fclose(F_INPUT)
sumbj := sum(j in J0) b(j)

fopen(JmnPocKNV,F_INPUT)
   readln
   forall(i in 1..PocZak) readln(c(i))
fclose(F_INPUT)

! Reading of noJ
fopen(Jmn_noJ,F_INPUT)
   readln
   forall(k in 1..noU) readln(noJ(k))
fclose(F_INPUT)


! Reading of f
fopen(Jmn_f,F_INPUT)
   readln
   forall(i in I0) readln(f(i))
fclose(F_INPUT)

! Reading of u
fopen(Jmn_u,F_INPUT)
   readln
   forall(i in I0) readln(u(i))
```

```
fclose(F_INPUT)


! Reading of sJ
fopen(Jmn_sJ,F_INPUT)
   readln
   readln
   forall(k in 1..noU,j in 1..TT) do
      readln(sJ(k,j))
      !sJ(k,j)+=1
   end-do
fclose(F_INPUT)

! Reading of q
fopen(Jmn_qH,F_INPUT)
   readln
   readln
   forall(k in 1..noU,j in 1..r) do
      readln(q(k,j))
   end-do
fclose(F_INPUT)

! Reading of the names of the dwelling places

fopen(JmnNazvyObci, F_INPUT)
  readln !Pocet moznych umisteni
  forall(j in J0) readln(NodeName(j))
fclose(F_INPUT)

! Reading of the current deployment of facilities
fopen(JmnSucY, F_INPUT)
  readln !Pocet uzivatelu
  forall(j in I0) readln(ff(j))
fclose(F_INPUT)

fopen(JmnGPS,F_INPUT)
   readln
   forall(i in 1..n) readln(lat(i),lng(i))
fclose(F_INPUT)

! Test of input noJ
writeln
forall(k in 1..noU) writeln(" noJ ",noJ(k))

! Test of input JB
writeln
forall(k in 1..noU) writeln(" JB ",sum(i in 1..noJ(k))b(sJ(k,i)))

! Test of input sJ
writeln
forall(k in 1..noU) do
   write(" sJ ", k," : ")
```

```
   forall(j in 1..30) write(strfmt(sJ(k,j),5))
   writeln
end-do

! Test of input qH into q
writeln
forall(k in 1..noU) do
   write(" q ", k," : ")
   forall(j in 1..r) write("  ",strfmt(q(k,j),10,8))
   writeln
end-do


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  End of reading input data
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Decision of meaning of weights w (either b or c)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

 if(WW="b") then
   forall(i in 1..PocZak) w(i):=b(i)
else
          if (WW="c") then
             forall(i in 1..PocZak) w(i):=c(i)
          end-if
end-if

!JmnOut:=JmnOut+"_No"+StrnoExp+WW+"_LZ_"+dlzkaZony+".txt"

!-------------------------------------------------------------------------------------


!GLOBALNE PREMENNE

declarations
 celkovyCas: real               !Celkovy cas (vsetky vypocty dokopy)
 cas: real                    !Cas jedneho vypoctu
 vysledokProcedury: real          !Navratova hodnota procedury
 approxObjFval: real             !Ucelova funkcia modelu (approx)
 realObjFval: real              !Prepocet na konvexnej kombinacii matic
 basicObjFval: real              !Prepocet na basic scenari
 zoznamStanic: array(1..p) of integer !Zoznam umiestnenych stanic

end-declarations

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Test of the approaches radial and allocation
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

declarations
```

```
        m:integer
        sI1:array(1..p) of integer
        sF1:array(1..p) of integer
        sI2:array(1..p) of integer
        sF2:array(1..p) of integer
        sI3:array(1..p) of integer
        sF3:array(1..p) of integer

        noI1:integer ! number of centers in sI1 and sF1
        noI2:integer ! number of centers in sI2 and sF2
        noI3:integer ! number of centers in sI3 and sF3

        RadObjavg:real
        RadFncavg:real
        LocObjavg:real
        LocFncavg:real
        CurFncavg:real
        CompTimeRad:real
        CompTimeLoc:real

  R1 = 1..r+1
        R0 = 1..r
        minr:array(R1,J0) of integer
        iY:array(I0) of integer
        k:integer

        invY:array(I0)of integer ! an inversion of sY to 1..p
        !VyslFNCGPH:real
end-declarations

!KONSTANTY

! setting on of the input parameters for getWPMGPrad23
m:=41 !pre dlzku zony 1
while(m*dlzkaZony < 41)
m := m + 1


! For debugging only
!forall(i in I0) f(i):=0
N:=1
B:=0
D:=m

forall(j in J0) w(j):=b(j)

writeln
writeln
writeln("NumberOfExperiment ",noExp)
!writeln ("Experimens for beta  ",beta)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Radialni pristup
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

startTime := gettime
RadObjavg:=getWPMGPHrad25avg(I0, J0, r, p, m,N,B,D,
                    noU, TT,
                    q,w, w, dij,
                    noJ, sJ, f,u,dlzkaZony,
                    sI1, sF1)
CompTimeRad := gettime - startTime

noI1:=0
forall(i in 1..p|sI1(i)>0.1) noI1+=1

RadFncavg:=FNCGPHavg(
          I0, J0, R0, p, r,noU,
          q,
          dij,
          noJ, ! noJ(u) the number of elements of the part u
          sJ, ! sJ(u) the list of locations of the part u
          w,
          noI1,
          sI1,!sY
          sF1,
          iY,
          minr)

writeln
writeln("Test of Radial approach for dlzkuZony  ",dlzkaZony)
writeln
writeln("RadObjavg ",RadObjavg)
writeln("CompTimeRad  ",CompTimeRad)
writeln("RadFncavg ",RadFncavg)
writeln
writeln("Vypis umisteni")
forall(i in 1..noI1) writeln(" sI1(",strfmt(i,3),")= ",strfmt(sI1(i),3),"b(sI1(i)) ",
strfmt(w(sI1(i)),10),"  sF1 ",strfmt(sF1(i),3),"  ",NodeName(sI1(i)))
writeln



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Alokacni pristup
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

startTime := gettime
LocObjavg:=getWPMGPHloc25avg(I0, J0, r, p, m,N,B,D,
                    noU, TT,
                    q,w, w, dij,
                    noJ, sJ, f,u,
                    sI2, sF2)
CompTimeLoc := gettime - startTime
```

```
noI2:=0
forall(i in 1..p|sI2(i)>0.1) noI2+=1

LocFncavg:=FNCGPHavg(
        I0, J0, R0, p, r,noU,
        q,
        dij,
        noJ, ! noJ(u) the number of elements of the part u
        sJ, ! sJ(u) the list of locations of the part u
        w,
        noI2,
        sI2,!sY
        sF2,
        iY,
        minr)

writeln
writeln("Test of Alocation approach")
writeln
writeln("LocObjavg ",LocObjavg)
writeln("CompTimeLoc  ",CompTimeLoc)
writeln("LocFncavg ",LocFncavg)
writeln
writeln("Vypis umisteni")
forall(i in 1..noI2) writeln(" sI2(",strfmt(i,3),")= ",strfmt(sI2(i),3),"w(sI2(i)) ",
strfmt(w(sI2(i)),10),"  sF2 ",strfmt(sF2(i),3),"  ",NodeName(sI2(i)))
writeln


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Vyhodnoceni soucasneho stavu
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Nacteni soucasneho stavu

fopen(JmnSucY,F_INPUT)
  readln
  forall(i in I0) readln(refY(i))
fclose(F_INPUT)

! refY transformujeme do noI3, sI3 a sF3
noI3:=0
forall(i in I0|refY(i)>0.1) do
  noI3+=1
  sI3(noI3):=i
  sF3(noI3):=refY(i)
end-do

! Vypocet ucelove funkce pro soucasny stav

CurFncavg:=FNCGPHavg(
        I0, J0, R0, p, r,noU,
```

```
        q,
        dij,
        noJ, ! noJ(u) the number of elements of the part u
        sJ, ! sJ(u) the list of locations of the part u
        w,
        noI3,
        sI3,!sY
        sF3,
        iY,
        minr)

writeln
writeln("Vyhodnoceni_"+ JmnReg+"_SucasnyStavYref.txt ")
writeln
writeln("CurFncavg ", CurFncavg)
writeln
writeln("Vypis umisteni")
forall(i in 1..noI3) writeln(" sI3(",strfmt(i,3),")= ",strfmt(sI3(i),3),"w(sI3(i)) ",
strfmt(w(sI3(i)),10),"  sF3 ",strfmt(sF3(i),3),"  ",NodeName(sI3(i)))
writeln

fopen(JmnOut, F_OUTPUT)
writeln
writeln
writeln("NumberOfExperiment ",noExp)
!writeln ("Experimens for beta  ",beta)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Radialni pristup
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
writeln
writeln("Test of Radial approach for dlzkuZony  ",dlzkaZony)
writeln
writeln("RadObjavg ",RadObjavg)
writeln("CompTimeRad  ",CompTimeRad)
writeln("RadFncavg ",RadFncavg)
writeln
writeln("Vypis umisteni")
forall(i in 1..noI1) writeln(" sI1(",strfmt(i,3),")= ",strfmt(sI1(i),3),"w(sI1(i)) ",
strfmt(w(sI1(i)),10),"  sF1 ",strfmt(sF1(i),3),"  ",NodeName(sI1(i)))
writeln

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Alokacni pristup
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

writeln
writeln("Test of Alocation approach")
writeln
writeln("LocObjavg ",LocObjavg)
writeln("CompTimeLoc  ",CompTimeLoc)
writeln("LocFncavg ",LocFncavg)
writeln
```

```
writeln("Vypis umisteni")
forall(i in 1..noI2) writeln(" sI2(",strfmt(i,3),")= ",strfmt(sI2(i),3),"w(sI2(i)) ",
strfmt(w(sI2(i)),10),"  sF2 ",strfmt(sF2(i),3),"  ",NodeName(sI2(i)))
writeln

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!  Vyhodnoceni soucasneho stavu
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


writeln
writeln("Vyhodnoceni_"+ JmnReg+"_SucasnyStavYref.txt ")
writeln
writeln("CurFncavg ", CurFncavg)
writeln
writeln("Vypis umisteni")
forall(i in 1..noI3) writeln(" sI3(",strfmt(i,3),")= ",strfmt(sI3(i),3),"w(sI3(i)) ",
strfmt(w(sI3(i)),10),"  sF3 ",strfmt(sF3(i),3),"  ",NodeName(sI3(i)))
writeln

fclose(F_OUTPUT)
(!
fopen(JmnVystupProtokol + '_' + strfmt(dlzkaZony, 0, 2) + '.txt', F_OUTPUT)
 writeln('pocKand: ', pocKand)
 writeln('pocZak: ', pocZak)
 writeln('sumbj: ', sumbj)
 writeln
 writeln('Dlzka zony: ', strfmt(dlzkaZony, 0, 2))
 writeln('Pocet premennych ', pocKand*pocZak*m)
 writeln('m: ', m)
 writeln('CompTime [s]: ', strfmt(compTime, 0, 2))
 writeln
 writeln('Optimalizacia - radial')
 writeln('Radial_refObj2: ', strfmt(RadObj, 0, 0))
 writeln('Radial_avgDist: ', strfmt(RadObj / sumbj, 0, 3))
 writeln
 writeln('Optimalizacia - real')
 writeln('OptSol_refObj2: ', strfmt(VyslFNCGPH, 0, 0))
 writeln('OptSol_avgDist: ', strfmt(VyslFNCGPH / sumbj, 0, 3))
 writeln
 writeln('Sucasny stav')
 writeln('SucStav_refObj2: ', strfmt(refObj2, 0, 0))
 writeln('SucStav_avgDist: ', strfmt(refObj2 / sumbj, 0, 3))
fclose(F_OUTPUT)

writeln
writeln
writeln
writeln('pocKand: ', pocKand)
writeln('pocZak: ', pocZak)
writeln('sumbj: ', sumbj)
writeln
```

```
writeln('Dlzka zony: ', strfmt(dlzkaZony, 0, 2))
writeln('Pocet premennych ', pocKand*pocZak*m)
writeln('m: ', m)
writeln('CompTime [s]: ', strfmt(compTime, 0, 2))
writeln
writeln('Optimalizacia - radial')
writeln('Radial_refObj2: ', strfmt(RadObj, 0, 0))
writeln('Radial_avgDist: ', strfmt(RadObj / sumbj, 0, 3))
writeln
writeln('Optimalizacia - real')
writeln('OptSol_refObj2: ', strfmt(VyslFNCGPH, 0, 0))
writeln('OptSol_avgDist: ', strfmt(VyslFNCGPH / sumbj, 0, 3))
writeln
writeln('Sucasny stav')
writeln('SucStav_refObj2: ', strfmt(refObj2, 0, 0))
writeln('SucStav_avgDist: ', strfmt(refObj2 / sumbj, 0, 3))
writeln
writeln("HOTOVO")
!)
end-model  ! InterregOpt25_191125
```