Многокритериальная оптимизация

Тушавин В. А.

8 декабря 2015 г.

Рассмотрим задачу. Рекламное агентство, в штате которого десять человек получило заказ на рекламу нового продукта на радио и телевидении. Данные о рекламной аудитории, стоимости ркуламы и количестве занятых при её изготовлении агентов заданы в таблице.

|  |  |  |
| --- | --- | --- |
| Характеристики | Радио | Телевидение |
| Рекламная аудитория (млн. чел) | 4 | 8 |
| Стоимость минуты рекламы ( в тыс. у.е.) | 8 | 24 |
| Количесво занятых агентов | 1 | 2 |

Сколько минут рекламного времени должно купить агентство на радио и ТВ, чтобы максимизировать аудитоию и минимизировать издержки, если контракт запрещает более 6 минут на радио?

Имеем следующую задачу:

Если решать задачу только на максимум имеем

library(lpSolve)   
f.obj <- c(4, 8) # Описали целевую функцию  
names(f.obj) <-c("X1","X2")  
a.mat<-rbind(c(1,0), # матрица  
 c(1,2), # коээфициентов  
 c(1,0), # при ограничениях  
 c(0,1))   
a.dir<-c("<=","<=",">=",">=")  
b.vec<-c(6,10,0,0) # вектор ограничений  
(result<-lp ("max", f.obj, a.mat, a.dir, b.vec))

## Success: the objective function is 40

result$solution

## [1] 0 5

Если решать задачу только на минимум

library(lpSolve)   
f.obj <- c(4, 24) # Описали целевую функцию  
names(f.obj) <-c("X1","X2")  
a.mat<-rbind(c(1,0), # матрица  
 c(1,2), # коээфициентов  
 c(1,0), # при ограничениях  
 c(0,1))   
a.dir<-c("<=","<=",">=",">=")  
b.vec<-c(6,10,0,0) # вектор ограничений  
(result<-lp ("min", f.obj, a.mat, a.dir, b.vec))

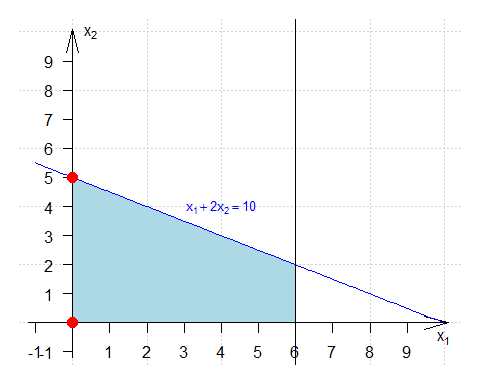
## Success: the objective function is 0

result$solution

## [1] 0 0

Графически задача выглядит следующим образом

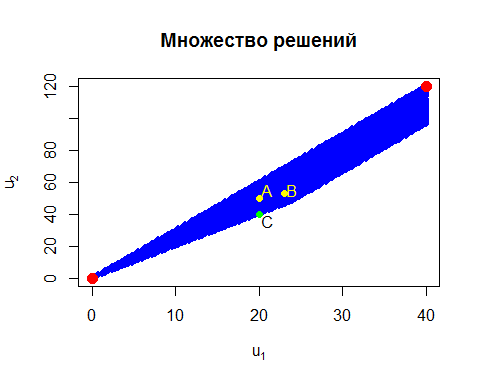
x1<- (-10:100)/10  
old<-par(mar=c(1,1,1,1))  
plot(0,type="n",xlab="",ylab="", xlim=c(-1, 10),ylim = c(-1, 10),bty="n",xaxt="n",yaxt="n")  
grid()  
polygon(c(0,0,6,6),c(0,5,2,0), col = "lightblue", border = NA)  
axis(1,pos=c(0,0),at=c(-1,1,2,3,4,5,6,7,8,9))  
axis(2,pos=c(0,0),las=2,at=c(-1,1,2,3,4,5,6,7,8,9))  
arrows(-1.2,0,10.1,0,angle=15)  
arrows(0,-1.2,0,10.1,angle=15)  
lines(x1,(10-x1)/2,col="blue")  
text(4,4,expression(х[1]+2\*х[2]==10),cex=0.8,col="blue")  
abline(v=6)  
text(0.5,10,expression(х[2]))  
text(10,-0.5,expression(х[1]))  
points(0,0,cex=1.5,col="red",pch=19)  
points(0,5,cex=1.5,col="red",pch=19)



par(old)

Множество всех решений может быть представлено:

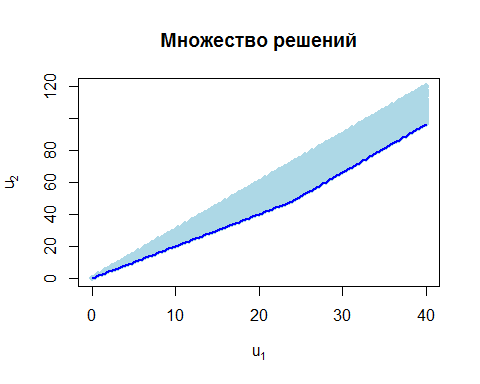
x1<-seq(0,6,by=0.1)  
x2<-seq(0,6,by=0.1)  
d<-expand.grid(x1=x1,x2=x2)  
d<-subset(d,x1+2\*x2<=10)  
d$u1<-4\*d$x1+8\*d$x2  
d$u2<-8\*d$x1+24\*d$x2  
plot(d$u2~d$u1,type="p",pch=19,col="blue",main="Множество решений",xlab=expression(u[1]),ylab=expression(u[2]),cex=0.8)  
points(0,0,cex=1.5,col="red",pch=19)  
points(40,120,cex=1.5,col="red",pch=19)  
points(20,50,cex=1,col="yellow",pch=19)  
text(21,55,"A",col="yellow")  
points(23,53,cex=1,col="yellow",pch=19)  
text(24,55,"B",col="yellow")  
  
points(20,40,cex=1,col="green",pch=19)  
text(21,36,"C")



Где u1 - аудитория в миллионах человек (эффективность), а u2 - стоимость рекламы. Рассмотрим решения А (20,50) и B(23,53). Вариант **A** имеет меньшую стоимость , но вариант **B** более эффективен. В таком случае можно говорить, что варианты несравнимы. Рассмотрим точку C(20,40). Как видим, при той же аудитории стоимость данного решения меньше.

Найдем меножество всех таких точек и построим на графике.

plot(d$u2~d$u1,type="p",pch=19,col="lightblue",main="Множество решений",xlab=expression(u[1]),ylab=expression(u[2]),cex=0.8)  
z<-aggregate(u1~u2,data=d,max)  
z<-aggregate(u2~u1,data=z,min)  
lines(z$u2~z$u1,col="blue",lwd=2)



Данная линия называется Парето-оптимальными вариантами.

### Метод идеальной точки

Метод идеальной точки (Метод Салуквадзе) состоит из двух этапов. На первом этапе находим наилучшее значение по всем критериям.

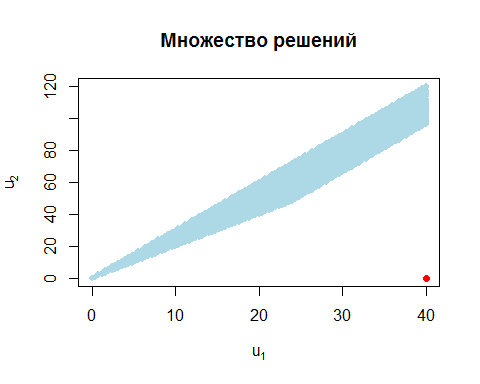
(u1<-max(d$u1))

## [1] 40

(u2<-min(d$u2))

## [1] 0

plot(d$u2~d$u1,type="p",pch=19,col="lightblue",main="Множество решений",xlab=expression(u[1]),ylab=expression(u[2]),cex=0.8)  
points(u1,u2,pch=19,col="red")



Данная точка u0 не принадлежит области допустимых решений. На втором этапе найдем решение, как точку, ближайшую к данной:

где R-- расстояние от u(X) до u0. В качестве R можно выбрать функцию:

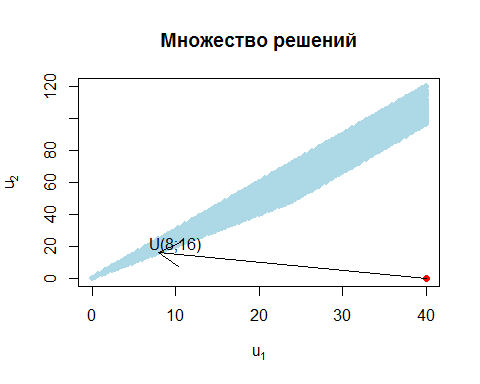
Произведем необходимые расчеты:

d.new<-d  
l<-2  
d.new$s<-((u1-d.new$u1)^l+(u2-d.new$u2)^l)^(1/l)  
(u.opt<-d.new[which.min( d.new$s),])

## x1 x2 u1 u2 s  
## 21 2 0 8 16 35.77709

Построим график

plot(d$u2~d$u1,type="p",pch=19,col="lightblue",main="Множество решений",xlab=expression(u[1]),ylab=expression(u[2]),cex=0.8)  
points(u1,u2,pch=19,col="red")  
arrows(u1,u2,u.opt$u1,u.opt$u2)  
text(u.opt$u1+2,u.opt$u2+5,paste0("U(",u.opt$u1,";",u.opt$u2,")"))



### Метод лексико-графического упорядочивания

На основании опроса ЛПР критерии ранжируются по важности. Предположим, что первым критерием мы выбираем охват аудитории. Тогда имеем множество возможных решений при максимальном критерии u1

(z<-max(d$u1))

## [1] 40

head(d.new<-subset(d,u1==z))

## x1 x2 u1 u2  
## 1281 6.0 2.0 40 96.0  
## 1340 5.8 2.1 40 96.8  
## 1399 5.6 2.2 40 97.6  
## 1458 5.4 2.3 40 98.4  
## 1517 5.2 2.4 40 99.2  
## 1576 5.0 2.5 40 100.0

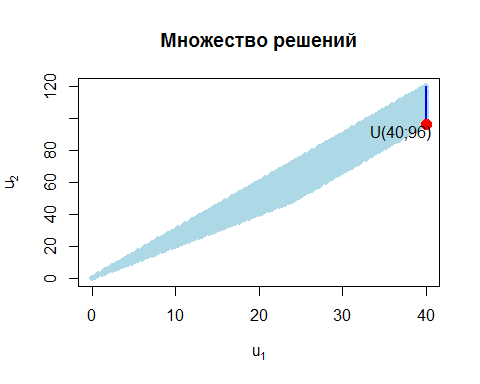
Находим наилучшее значение по второму критерию:

(u.opt<-d.new[which.min( d.new$u2),])

## x1 x2 u1 u2  
## 1281 6 2 40 96

Представим графически:

plot(d$u2~d$u1,type="p",pch=19,col="lightblue",main="Множество решений",xlab=expression(u[1]),ylab=expression(u[2]),cex=0.8)  
lines(d.new$u1,d.new$u2,col="blue",lwd=2)  
points(u.opt$u1,u.opt$u2,pch=19,col="red",cex=1.5)  
text(u.opt$u1-3,u.opt$u2-5,paste0("U(",u.opt$u1,";",u.opt$u2,")"))



### Метод линейной свертки

ЛПР задает значение весов критериев и решается задача максимизации критерия. Поскольку предполагается, что оба критерия должны быть максимизируемы, критерий u2 возьмем со знакоми минус. Пусть веса критериев равны 0.8 и 0.2, тогда:

d.new<-d  
d.new$s<-d.new$u1\*0.8-d.new$u2\*0.2  
head(d.new)

## x1 x2 u1 u2 s  
## 1 0.0 0 0.0 0.0 0.00  
## 2 0.1 0 0.4 0.8 0.16  
## 3 0.2 0 0.8 1.6 0.32  
## 4 0.3 0 1.2 2.4 0.48  
## 5 0.4 0 1.6 3.2 0.64  
## 6 0.5 0 2.0 4.0 0.80

tail(d.new)

## x1 x2 u1 u2 s  
## 2931 0.2 4.8 39.2 116.8 8.00  
## 2932 0.3 4.8 39.6 117.6 8.16  
## 2990 0.0 4.9 39.2 117.6 7.84  
## 2991 0.1 4.9 39.6 118.4 8.00  
## 2992 0.2 4.9 40.0 119.2 8.16  
## 3051 0.0 5.0 40.0 120.0 8.00

subset(d.new,s==max(d.new$s))

## x1 x2 u1 u2 s  
## 1281 6 2 40 96 12.8

Пусть веса критериев равны

d.new<-d  
d.new$s<-d.new$u1\*0.5-d.new$u2\*0.5  
head(d.new)

## x1 x2 u1 u2 s  
## 1 0.0 0 0.0 0.0 0.0  
## 2 0.1 0 0.4 0.8 -0.2  
## 3 0.2 0 0.8 1.6 -0.4  
## 4 0.3 0 1.2 2.4 -0.6  
## 5 0.4 0 1.6 3.2 -0.8  
## 6 0.5 0 2.0 4.0 -1.0

tail(d.new)

## x1 x2 u1 u2 s  
## 2931 0.2 4.8 39.2 116.8 -38.8  
## 2932 0.3 4.8 39.6 117.6 -39.0  
## 2990 0.0 4.9 39.2 117.6 -39.2  
## 2991 0.1 4.9 39.6 118.4 -39.4  
## 2992 0.2 4.9 40.0 119.2 -39.6  
## 3051 0.0 5.0 40.0 120.0 -40.0

subset(d.new,s==max(d.new$s))

## x1 x2 u1 u2 s  
## 1 0 0 0 0 0

Нетрудно увидеть, что фактически в данном случае мы сводим задачу к задаче линейного программирования. Пусть k1 и k2 весовые коэффициенты, тогда имеем.

Откуда:

при тех же ограничениях

Пусть k1=0.8, а k2=0.2

k1<-0.8  
k2<-0.2  
f.obj <- c(4\*k1-8\*k2, 8\*k1-24\*k2) # Описали целевую функцию  
names(f.obj) <-c("X1","X2")  
a.mat<-rbind(c(1,0), # матрица  
 c(1,2), # коээфициентов  
 c(1,0), # при ограничениях  
 c(0,1))   
a.dir<-c("<=","<=",">=",">=")  
b.vec<-c(6,10,0,0) # вектор ограничений  
(result<-lp ("max", f.obj, a.mat, a.dir, b.vec))

## Success: the objective function is 12.8

result$solution

## [1] 6 2

Результаты аналогичны вышеприведенному примеру.

#### Информация о параметрах R

sessionInfo()

## R version 3.2.2 (2015-08-14)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 8 x64 (build 9200)  
##   
## locale:  
## [1] LC\_COLLATE=Russian\_Russia.1251 LC\_CTYPE=Russian\_Russia.1251   
## [3] LC\_MONETARY=Russian\_Russia.1251 LC\_NUMERIC=C   
## [5] LC\_TIME=Russian\_Russia.1251   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] lpSolve\_5.6.13  
##   
## loaded via a namespace (and not attached):  
## [1] magrittr\_1.5 formatR\_1.2.1 tools\_3.2.2 htmltools\_0.2.6  
## [5] yaml\_2.1.13 stringi\_1.0-1 rmarkdown\_0.8.1 knitr\_1.11   
## [9] stringr\_1.0.0 digest\_0.6.8 evaluate\_0.8