Санкт-Петербургский Государственный Технический Университет

Факультет Технической Кибернетики

Кафедра Системный Анализ и Управление

#### ПРИНЯТИЕ РЕШЕНИЙ

### Расчетное задание

### Тема: "Задача об упаковке"

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_

### Санкт-Петербург

### 2001 г.

Содержание

1.Постановка задачи............................................................................................…….2

2.Теоретическая часть………………………………………………………………..3

3.Решение……………………………………………………………………………..5

4.Алгоритм программы………………………………………………………………6

5.Результаты…………………………………………………………………………..7

6.Выводы……………………………………………………………………………...7

Приложение…………………………………………………………………………..8

1.Постановка задачи.

Рассмотреть задачу об упаковке 20 гипотетических объектов в пять контейнеров. Объекты имеют оценки по пяти критериям Б,В,Г,Д,Е с порядковыми шкалами, имеющими три градации (первая - лучшая, вторая – средняя, третья - худшая), а также два физических параметра (вес и объем). Критерии имеют одинаковую значимость. Контейнеры имеют следующие параметры:

* Грузоподъемность контейнера – 5
* Объем контейнера – 7

Далее приведены данные объектов:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер | Вес | Обьем | Б | В | Г | Д | Е |
| 1 | 3 | 2 | 3 | 3 | 3 | 3 | 3 |
| 2 | 1 | 1 | 3 | 2 | 2 | 1 | 1 |
| 3 | 3 | 1 | 2 | 1 | 1 | 1 | 2 |
| 4 | 2 | 3 | 2 | 1 | 3 | 2 | 3 |
| 5 | 1 | 1 | 1 | 1 | 1 | 3 | 3 |
| 6 | 3 | 2 | 2 | 2 | 1 | 1 | 1 |
| 7 | 1 | 2 | 3 | 1 | 3 | 3 | 1 |
| 8 | 2 | 1 | 1 | 1 | 1 | 2 | 3 |
| 9 | 3 | 2 | 2 | 2 | 1 | 3 | 2 |
| 10 | 2 | 1 | 1 | 1 | 2 | 2 | 2 |
| 11 | 1 | 2 | 3 | 3 | 1 | 1 | 1 |
| 12 | 3 | 1 | 2 | 1 | 2 | 3 | 1 |
| 13 | 1 | 1 | 2 | 2 | 3 | 3 | 1 |
| 14 | 1 | 1 | 3 | 3 | 3 | 2 | 1 |
| 15 | 2 | 2 | 1 | 2 | 2 | 1 | 1 |
| 16 | 3 | 2 | 3 | 1 | 2 | 1 | 3 |
| 17 | 1 | 1 | 2 | 1 | 2 | 1 | 2 |
| 18 | 2 | 2 | 3 | 1 | 3 | 2 | 1 |
| 19 | 1 | 1 | 1 | 1 | 1 | 2 | 1 |
| 20 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

2.Теоретическая часть.

Рассмотрим следующую задачу: имеется множество из М объектов, которое желательно упаковать в К емкостей для последующей перевозки, причем М существенно больше К. Каждый объект характеризуется Р -количественными физическими параметрами (весом и объемом); каждая емкость характеризуется этими же предельными физическими параметрами (например, общим объемом и грузоподъемностью). Кроме того, каждый из упаковываемых объектов имеет оценки по нескольким критериям , которые характеризуют его качество и привлекательность для лица, ответственного за перевозку. Емкость контейнеров недостаточна для упаковки всех имеющихся объектов. Желательно осуществить упаковку наилучшим образом, т.е. так чтобы:

1. Число упакованных объектов было бы максимально возможным, так как все они в той или иной степени заслуживают упаковки в емкости (т.е. предварительный отбор, исключающий абсолютно плохие объекты, уже сделан) – критерий О1.
2. Среди упакованных объектов было бы наибольшее количество таких, качество которых превосходило бы качество неупакованных – критерий О2.

Имеется конечное множество объектов, причем размер каждого из них задан рациональным числом. Требуется упаковать предметы в минимально возможное количество контейнеров так, чтобы суммарный размер объектов в каждом контейнере не превышал его размер (также рациональное число).

Для решения этой задачи предлагается ряд алгоритмов:

1. Алгоритм "в первый подходящий". Пусть имеется какой-то порядок объектов и контейнеров. Первый предмет кладем в первый попавшийся контейнер. Второй объект кладем в первый контейнер, если он туда помещается, а если нет – то во второй контейнер. Аналогично упаковываем прочие объекты.
2. Алгоритм "в первый подходящий с убыванием". Упорядочим список объектов от больших к меньшим. Далее используем алгоритм "в первый подходящий".
3. Алгоритм "в лучший из подходящих". Пусть имеется какой-то произвольный порядок объектов. Идея упаковки аналогична алгоритму "в первый подходящий", но со следующей разницей: очередной объект кладется в тот контейнер, где имеется наименьшее, но достаточное для него неиспользованное пространство.
4. Алгоритм "в лучший из подходящих с убыванием". Алгоритм аналогичен "в лучший из подходящих", но объекты упорядочены от больших к меньшим.

Упаковываемые объекты имеют оценки качества по многим критериям. Требуется упаковать максимальное число объектов, а не получить минимальное число контейнеров. Введем следующие обозначения:



vij – j-й физический параметр i-го объекта;

Vlj – j-й физический параметр l-го контейнера;

Ui – общая ценность i-го объекта.

Обозначим через I=1, 2, …, М множество номеров объектов, а через



Множество тех упакованных объектов, для которых не найдется более ценных среди не упакованных. Формальная постановка задачи имеет следующий вид:

, .

При ограничениях:

, j = 1, …, P, l = 1, …, K; 

Алгоритм решения поставленной задачи включает в себя алгоритмы решения вспомогательных задач:

1.Упаковка многомерных объектов в контейнеры;

2.Получение информации от ЛПР, позволяющей определить порядок упаковки многокритериальных объектов.

3.Решение задачи.

1. Путем попарного сравнения упаковываемых объектов определяется асимметричное транзитивное отношение доминирования:



где Q – количество критериев.

1. В соответствии с отношением P0 на множестве упаковываемых объектов можно выделить подмножество недоминируемых объектов. После их удаления можно выделить второе подмножество и т.д. до исчерпания множества. Выделенные подмножества называются паретовыми слоями.
2. Применяем алгоритм упаковки с отбрасыванием при чередовании, упаковывая по слоям объекты в контейнеры.

К построенному квазипорядку упаковываемых объектов итеративно применяем алгоритм упаковки с отбрасыванием для послойной упаковки объектов. Пусть объекты первых (i–1)-го слоев упаковываются, а элементы i слоев не упаковываются. Среди объектов, входящих в первые (i=1) слои, выделяется подмножество лучших объектов, превосходящих каждый из остальных упаковываемых объектов (если таковое имеется). Это подмножество считается на данном этапе решения задачи подлежащим обязательной упаковке. Получим одну из возможных упаковок, наилучших с точки зрения О2.

Будем упаковывать, используя алгоритм с отбрасыванием при чередовании, объекты первых i слоев. Последовательно удаляем при упаковке объекты i-го слоя в соответствии с их порядком в списке с чередованием (от первых к последним) до получения упаковки. Если при этом в контейнерах остаются свободные места по всем физическим параметрам, то в рассмотрение включаются объекты (i+1)-го слоя, недоминируемые неупакованными объектами i-го слоя, и осуществляется доупаковка. Если и теперь остаются возможности, то аналогично осуществляется упаковка некоторыми объектами (i+2)-го слоя и т.д. В итоге получаем упаковку с максимальным значением критерия О2.

Получим теперь упаковку с максимальным значением критерия О1.

Применим алгоритм АОЧ ко всему множеству упаковываемых объектов. Не удаляются только упомянутые выше наилучшие объекты, доминирующие по качеству над всеми остальными (если таковые имеются). Ясно, что при этом получим упаковку с максимальным значением критерия О1 при условии сохранения доминирующих объектов. Рассматривая точки на плоскости О1 – О2, ЛПР определить наилучший для себя компромисс между критериями О1 и О2 и тем самым наилучшую упаковку.

4.Алгоритм программы.

Начало

Инициализация данных

Разбиение на пар. слои

Сорт. объем /вес

Упаковка по слоям

Упаковка объем/вес

Вывод

Конец

5.Результаты.

После выполнения программы получены следующие результаты.

Программа распределила объекты из исходного списка по паретовым слоям.

Ниже приведены эти слои (в таблице указаны номера эл-тов):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Слой | Номера объектов | | | | | | |
| 1 | 20 |  |  |  |  |  |  |  |
| 2 | 3 | 6 | 15 | 19 |  |  |  |  |
| 3 | 2 | 8 | 9 | 10 | 11 | 12 | 17 | 18 |
| 4 | 4 | 5 | 7 | 13 | 14 | 16 |  |  |
| 5 | 1 |  |  |  |  |  |  |  |

Далее программа отсортировала список объектов по очередности макс.вес /

макс.объем.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 3 | 6 | 9 | 7 | 12 | 16 | 11 | 15 | 8 | 10 | 18 | 20 | 2 | 5 | 13 | 14 | 17 | 19 |

Ниже приведена таблица результатов упаковки (по алгоритму упаковки с отбрасыванием).

|  |  |
| --- | --- |
| Кол-во | Σ Польза |
| 14 | 123 |
| 10 | 83 |

Результаты можно отразить графически в виде плоскости критериев О1(суммарное количество упакованных предметов), О2(суммарная полезность упакованных элементов).



6.Выводы.

В результате выполнения задания была написана программа, упаковывающая объекты в контейнеры. Упаковка производится с помощью двух вариантов упорядочивания объектов. По критерию О1(кол-во упакованных) наиболее эффективен второй метод(есть варианты упаковки по 14 предметов). Например, были упакованы следующие 14 предметов:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 16 | 11 | 15 | 8 | 10 | 18 | 20 | 2 | 5 | 13 | 14 | 17 | 19 | 7 |

О1 =14, О2 =130.

По критерию О2 выигрывает первый метод.

Упакованные объекты:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 14 | 16 | 1 | 20 | 3 | 6 | 15 | 19 | 2 | 8 |

О1 =10, О2 =83.

Оба метода позволяют ЛПР выбрать оптимальный вариант упаковки на плоскости критериев О1 ,О2.

Приложение.

Текст программы.

Программа написана на языке программирования С++ в среде разработки Visual C++ 6.0. Выбор языка обусловлен наличием в его стандарте структуры данных – класс, с помощью которой удобно моделировать объекты задания.

#include <stdlib.h>

#include <fstream.h>

#include "iostream.h"

#include "stdio.h"

class Object{

public:

int Mass;

int Cap;

int vol[5];

int Val;

bool Packed;

int INN;

bool NDom;

Object(){

Mass = 0;

Cap = 0;

Packed = false;

vol[0] = 0;

vol[1] = 0;

vol[2] = 0;

vol[3] = 0;

vol[4] = 0;

Val=0;

INN=0;

NDom=false;

};

void ObjectInit(int m, int c, int v1, int v2, int v3, int v4, int v5,int inn)

{

Mass=m;

Cap=c;

Packed=false;

vol[0]=v1;

vol[1]=v2;

vol[2]=v3;

vol[3]=v4;

vol[4]=v5;

Val= vol[0]+vol[1]+vol[2]+vol[3]+vol[4];

INN=inn;

NDom=false;

};

};

class Konteiner{

public:

int Mass;

int Cap;

Konteiner(){

Mass = 0;

Cap = 0;

};

void KonteinerInit(int m, int c){

Mass = m;

Cap = c;

};

};

struct Result{

int Value;

int Num;

};

void main(){

Object Ob[21],ObD[400],ObND[400],ObRs,ObMC1[20],ObMC2[20],ObMC[21],ObMCRs[20];

Object ObSlice[10][10];

bool MFLAG[21];

Result Res[20],Res1[20];

Konteiner Kon[5];

Ob[0].ObjectInit(3,2,3,3,3,3,3, 1);

Ob[1].ObjectInit(1,1,3,2,2,1,1, 2);

Ob[2].ObjectInit(3,1,2,1,1,1,2, 3);

Ob[3].ObjectInit(2,3,2,1,3,2,3, 4);

Ob[4].ObjectInit(1,1,1,1,1,3,3, 5);

Ob[5].ObjectInit(3,2,2,2,1,1,1, 6);

Ob[6].ObjectInit(1,2,3,1,3,3,1, 7);

Ob[7].ObjectInit(2,1,1,1,1,2,3, 8);

Ob[8].ObjectInit(3,2,2,2,1,3,2, 9);

Ob[9].ObjectInit(2,1,1,1,2,2,2,10);

Ob[10].ObjectInit(1,2,3,3,1,1,1,11);

Ob[11].ObjectInit(3,1,2,1,2,3,1,12);

Ob[12].ObjectInit(1,1,2,2,3,3,1,13);

Ob[13].ObjectInit(1,1,3,3,3,2,1,14);

Ob[14].ObjectInit(2,2,1,2,2,1,1,15);

Ob[15].ObjectInit(3,2,3,1,2,1,3,16);

Ob[16].ObjectInit(1,1,2,1,2,1,2,17);

Ob[17].ObjectInit(2,2,3,1,3,2,1,18);

Ob[18].ObjectInit(1,1,1,1,1,2,1,19);

Ob[19].ObjectInit(1,2,1,1,1,1,1,20);

for (int i=0;i<5;i++){

Kon[i].KonteinerInit(5,7);

};

MFLAG[0]=true;

for(i=1;i<21;i++){

MFLAG[i]=false;

};

bool flag,superflag;

superflag=true;

int counter=0;

int j;

while(counter!=10){

superflag=false;

for(i=0;i<200;i++){ObND[i].ObjectInit(0,0,0,0,0,0,0,0);ObD[i].ObjectInit(0,0,0,0,0,0,0,0);};

j=0;

for(int l=0;l<20;l++){

for(i=0;i<20;i++){

if((MFLAG[Ob[i].INN]==false)&&(MFLAG[Ob[l].INN]==false)&&(i!=l)&&(Ob[l].vol[0]>=Ob[i].vol[0])&&(Ob[l].vol[1]>=Ob[i].vol[1])&&(Ob[l].vol[2]>=Ob[i].vol[2])&&(Ob[l].vol[3]>=Ob[i].vol[3])&&(Ob[l].vol[4]>=Ob[i].vol[4])){

ObD[j]=Ob[l]; ObND[j]=Ob[i];j++;}else{

if((MFLAG[Ob[i].INN]==false)&(MFLAG[Ob[l].INN]==false)&&(i!=l)&&(Ob[l].vol[0]<=Ob[i].vol[0])&&(Ob[l].vol[1]<=Ob[i].vol[1])&&(Ob[l].vol[2]<=Ob[i].vol[2])&&(Ob[l].vol[3]<=Ob[i].vol[3])&&(Ob[l].vol[4]<=Ob[i].vol[4])){

ObD[j]=Ob[i]; ObND[j]=Ob[l];j++;};

};

};

};

j=0;

for(l=0;l<200;l++){

flag=true;

for(i=0;i<200;i++){

if(ObND[l].INN==ObD[i].INN){flag=false;};

};

if(flag&&(MFLAG[ObND[l].INN]!=true)){ObSlice[counter][j]=ObND[l];MFLAG[ObND[l].INN]=true;j++;};

};

counter++;

};

for(counter=0;counter<10;counter++){

if(ObSlice[counter][0].INN==0){ObSlice[counter][0]=Ob[0];break;};

for( i=0;i<20;i++){ ObMC1[i] = Ob[i];};

for( j=0;j<20;j++){

for(i=19;i>j;i--){

if((ObMC1[i-1].Mass<ObMC1[i].Mass)){

ObRs=ObMC1[i]; ObMC1[i]=ObMC1[i-1]; ObMC1[i-1]=ObRs;

};

};

};

for(i=0;i<20;i++){

ObMCRs[i]=ObMC1[i];

};

for(i=0;i<20;i++){

cout<<ObMCRs[i].INN<<" ";

};

for( i=0;i<20;i++){ ObMC2[i] = Ob[i];};

for( j=0;j<20;j++){

for(i=19;i>j;i--){

if((ObMC2[i-1].Cap<ObMC2[i].Cap)){

ObRs=ObMC2[i]; ObMC2[i]=ObMC2[i-1]; ObMC2[i-1]=ObRs;

};

};

};

cout<<"\n";

for(i=0;i<20;i++){

cout<<ObMC2[i].INN<<" ";

};

flag=true;

bool flag1=true;

int n;

int m=0;

for(n=0;n<20;n++){

flag1=true; flag=true;

for(j=0;j<20;j++){

if((ObMCRs[n].INN==ObMC2[n].INN)||(ObMCRs[n].INN==ObMC[j].INN)){flag1=false;};

};

for(j=0;j<20;j++){

if(ObMC2[n].INN==ObMC[j].INN){flag=false;};

};

if((flag1)&&(flag)){

ObMC[m]=ObMCRs[n];

ObMC[m+1]=ObMC2[n];

m=m+2;

};

if((flag1)&&(!flag)){

ObMC[m]=ObMCRs[n];

m++;

};

if((!flag1)&&(flag)){

ObMC[m]=ObMC2[n];

m++;

};

if((!flag1)&&(!flag)){

};

};

cout<<"\n";

for(i=0;i<20;i++){

cout<<ObMC[i].INN<<" ";

};

int l=0;

m=0;

flag=true;

int countj=0;

int counti=0;

int lasti=0;

int Value=0;

int Num=0;

int count=0;

int countp=0;

for(j=0;j<10,ObSlice[j][0].INN!=0;j++){

for(i=0;i<10,ObSlice[j][i].INN!=0;i++){

Ob[count]=ObSlice[j][i];count++;};

};

count=0;

while ((count!=20)){

for(j=0;j<20;j++){

flag=true;

for(m=0;m<5;m++){

if(flag&&(Ob[j].Cap<Kon[m].Cap)&&(Ob[j].Mass<Kon[m].Mass)){

Kon[m].Cap=Kon[m].Cap-Ob[j].Cap;

Kon[m].Mass=Kon[m].Mass-Ob[j].Mass;

Value=Value+Ob[j].Val;

Num=Num++;

Ob[j].Packed=true;

flag=false;

};

};

};

Ob[20]=Ob[0];

for(i=1;i<21;i++){Ob[i-1]=Ob[i];};

Res[count].Value=Value;

Res[count].Num=Num;

if(count==0){

cout<<"\n";

for(i=0;i<20;i++){

if(Ob[i].Packed){cout<<Ob[i].INN<<" ";};

};

};

count++;

for(j=0;j<10;j++){

Ob[j].Packed=false;

};

Value=0;

Num=0;

for(m=0;m<5;m++){

Kon[m].KonteinerInit(5,7);

};

};

cout<<"\n";

flag=true;

countj=0;

counti=0;

lasti=0;

Value=0;

Num=0;

count=1;

countp=0;

while ((countj!=20)){

for(j=0;j<20;j++){

flag=true;

for(m=0;m<5;m++){

if(flag&&(ObMC[j].Cap<Kon[m].Cap)&&(ObMC[j].Mass<Kon[m].Mass)){

Kon[m].Cap=Kon[m].Cap-ObMC[j].Cap;

Kon[m].Mass=Kon[m].Mass-ObMC[j].Mass;

Value=Value+ObMC[j].Val;

Num++;

ObMC[j].Packed=true;

flag=false;

};

};

};

ObMC[20]=ObMC[0];

for(j=1;j<21;j++){ObMC[j-1]=ObMC[j];};

if(countj==8){

cout<<"\n";

for(i=0;i<20;i++){

if(ObMC[i].Packed){cout<<ObMC[i].INN<<" ";};

};

};

for(j=0;j<20;j++){

ObMC[j].Packed=false;

};

Res1[countj].Value=Value;

Res1[countj].Num=Num;

countj++;

Value=0;

Num=0;

for(m=0;m<5;m++){

Kon[m].KonteinerInit(5,7);

};

};

ofstream out("out.txt",ios::out|ios::trunc);

out<<" Итоговые данные после упаковки: \n";

out<<"Сортировка по Пар. сл.: Сортировка вес.объем:\n";

out<<"Ценность Кол-во Ценность Кол-во \n";

for(i=0;i<20;i++){

cout<<Res[i].Value<<" "<<Res[i].Num<<" ";

cout<<Res1[i].Value<<" "<<Res1[i].Num<<" \n";

out<<Res[i].Value<<" "<<Res[i].Num<<" ";

out<<Res1[i].Value<<" "<<Res1[i].Num<<" \n";

};

char ch;

cout<<"Press a key\n";

cin>>ch;

}