



Resource allocation algorithm

Zhu Lida, Yuan Jun

Supervisor: Nouredine Bouhmala

Project report for Web-mining in Autumn 2006

Agder University College
Faculty of Engineering and Science
Grimstad, 22 August 2006
Status: Draft

Keywords: page rank, K-clustering

Abstract:

The resource of the World Wide Web is almost infinite. But when we access into the Internet, some of the resource is easy to download, and some of the resource is hard to access or download. So it comes to the questions that which one is available resource, and which one is faster to access and download, and so on. Resource allocation is a problem of allocation your limited resources to fit your needs. [1] If we test the resource by the speed of access and download, it will be a kind of standard of evaluating the resource. But we can also consider the updated page as another kind of standard. So we can consider the both standards to evaluate the resource. When we have those multilevel techniques to evaluate the resource, it comes to the question about Knapsack problem which is a popular problem been used to solve the clustering problem. So in this project, we are going to find out which one is the best standard of clustering the random data of resource.

Version Control

Version	Status	Date	Change	Author
1.0	draft	22.11.06		Yuan Jun
1.0	draft	23.11.06		Zhu Lida
1.0	final	26.11.06		all

Table of Contents

Resource allocation algorithm	1
Abstract:.....	2
Version Control.....	3
Table of Contents.....	4
1. Executive summary.....	6
1.1 How is our combination algorithm work?	6
2. Introduction.....	7
2.1 Problem motivation.....	7
2.2 Knapsack problem and multilevel technique.....	7
2.3 Proposal outline	8
3. Background.....	8
3.1 What is K-clustering?	9
3.2 Multilevel techniques.....	9
3.3 Stochastic	11
4. Problem description	11
5. Requirement specification for K-clustering & multilevel techniques	12
6. Design of combination of K-clustering and multilevel techniques	13
6.1 List of Knapsack problems	13
6.2 algorithms of Knapsack problem.....	13
6.2.1 Value Density (Greedy solution)	14
6.2.2 Dynamic Programming	14
6.2.3 Optimal Binary Search Trees.....	14
6.3 Multilevel strategy	15
6.4 flow chart of program	16
7. Implementation	18
7.1 Failed with Dynamic programming.....	18
7.2 Implement in Greedy algorithm.....	18
8. Evaluation and testing.....	21
8.1 Coding.....	21
8.1.1 Testing.....	21
8.1.2 Evaluation	21
8.2 results	21
8.2.1 The first testing	21
8.2.2 The second testing.....	22
8.2.3 Third.....	23
8.2.4 Fourth.....	24
8.2.5 Final testing.....	26
9. Discussion	32
9.1 Project outcomes.....	32

9.1.1 What we did in this project?	32
9.1.2 What we can do in the future?	32
9.2 Evaluation of the overall results	33
10. Conclusion	34
Appendix.....	35
A1 References.....	35

1. Executive summary

In our regular life we receive a lot of information every day and some of them are valuable, therefore we save it as data-base for reference. But since the information's growth is very fast, maybe one week or two this data-base is become numerous, if we don't find a way to make it easy to handle and manage, we may have a lot of trouble when we reference later. And this problem also exists in web source. We know that web source is also huge so we have to find an easy way to make our searching more effective. We can see reference in data-base or search something on the web has the same problem is that we are face is really huge object which is far more than we can imagine.

So we need to find a way to solve this problem, as we thought we just need to find an algorithm which can automatic divide different object into different subset. But we come to the question that how to visualize this? So in our project we will introduce a combination of the multilevel technique and K-clustering. Multilevel technique is a kind of process which diving large and difficult problem into smaller one, which could make it much easier to handle. And K-clustering is a cluster way of cluster, and in this way of cluster the number K is very important.

In our work we have a data-base in local computer as our object and we find that each object has different traits, but dose all the objects have the same traits? Or does every trait have the same weight? Unfortunately, the answer is no. Therefore we can divide them into different subset. The entire object in this subset has the same trait, and this trait also has the highest value compared with others, in this way we can divide our object into different parts. So this could make our data-base much easier to handle and manage.

As we see our way of division is very familiar with another problem—knapsack problem. Thus we can use the solution of knapsack problem to solve our problem we now face. And in knapsack problem we have binary and fractional. Because each data object have many traits, so we should choose fractional solution in our algorithm.

1.1 How is our combination algorithm work?

Because we combine K-clustering and multilevel technique in algorithm, and use 1000 data as our objects, in the final result we can have a tree structure. In the lowest part of this structure we use the solution of fractional knapsack to get several data which is a better answer, and we will drop those data don't fit our solution. Then we

take these data to next higher level, and do the same work as we did. This work will keep going until we arrive the highest point. This progress will be done in each branch of the tree map, and finally we can get those data in the highest point and these data is the solution we looking for.

But in our life when we download a webpage we can not just down load parts of it. The choice we have is download all of it or not. This is means in our application we have to use binary knapsack as our solution. But in the course of judging a webpage's value we still need use fractional knapsack help us.

2. Introduction

2.1 Problem motivation

We know in our real life if we try to download something from a webpage that we know nothing about it. If we download the entire data it may cause a big "traffic jam" to our computer. Dealing with the web is particularly fascinating because it leads to numerous extremely interesting problems involving real-life resource allocation and scheduling [2].

In this project we will combine multilevel technique and K-clustering, and apply this in data-base management and web resource allocation so that we can evaluate a web page's value. Our goal is to make data-base and web resource is easy to handle and manage, let our visitation of data-base or searching information on internet is more effective.

We will give an automatic way (algorithm) to evaluate a web's value, let our computer or searching engine to give us the best webpage we need.

2.2 Knapsack problem and multilevel technique

Knapsack problem can describe like this, a thief goes to a house with a knapsack, and this thief can take any thing he wants. But his knapsack has a limit of weight, so he has to deal with this problem that under the maximum weight take the highest value. In this problem we have two solutions binary and fractional. Binary knapsack (same called integer knapsack as follow) is this thief can take each thing 100% or 0%, but the fractional is you can chose any percentage of each thing you want to put in his knapsack.

Multilevel technique is a kind of process to divide large and difficult problem into smaller ones, which are hopefully much easier to handle. When we use this way to deal a problem we usually get a tree structure.

Contribution of this paper: we try to find the feasibility of combine K-clustering and multilevel technique, and prove this combination work have some advantages than other similar work.

2.3 Proposal outline

In this paper we first give a briefly report of our project in chapter 1; then we will tell you our motivation and the two main techniques-multilevel and K-clustering, we used in our project. In 3rd chapter we will give more details about the technique we use in this project and how we use it in our combination work. In chapter 4 we will introduce the exactly course of how our design work. In 5th chapter is requirement description of these two techniques. And in the next chapter-chapter 6, we tell you our design's structure and something about our programming language-JAVA. In 7th chapter is more about how we implement our work. We will also give you some more detail about our evaluation and testing result. Before our conclusion we will show some discussion about this project, and how we can use it in application area. Finally we will review this project and also predict how is our work can be used.

3. Background

In our regular life we have a lot of valuable information, we save then as data. Some data are saved in local service and some are saved on internet as web source. Day and day these information grows bigger and bigger, and finally it will become numerous.

So when we want search some data from this incredible big data-base, it is really hard work. Therefore we need to find an easy way for our searching and make it more effective.

We know that each data has its own traits. And some of them are important but others may not. As we thought we can use data clustering to divide them into several groups by its different traits. And in order to make it more effective a multilevel schema will be used for our clustering problem which is K-clustering problem. Multilevel techniques refer to the process of dividing large and difficult problem into smaller

ones, which are hopefully much easier to handle and manage. And in each level we have a K to show how many subset we have in this level.

As we see when we divide each data-base into different groups, and this clustering way is similar with knapsack problem. Therefore we can use the solution of knapsack problem to solve our clustering problem.

3.1 What is K-clustering?

“The K-means algorithm assigns each point to the cluster whose center (also called centroid) is nearest. The center is the average of all the points in the cluster — that is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster.”[3]

Through this we can know that k-clustering is a kind of partition clustering, and each cluster has a centroid.

The number of cluster- K must be specified.

Data clustering is a common technique for data analysis, which we can use in many ways like data mining, machine learning, and pattern recognition [3]. And clustering is the classification of similar objects into different group (part), and those data in each group (part) share some common trait. In this project we will use multilevel techniques for k-clustering problems, which means we will use it to divide very large and complex data into different level and different small parts, which make it easier to handle and manage.

So, the first step we need to do is to use k-clustering to divide the similar data into a small group which is in the lowest level-level 1 of our multilevel schema. Therefore, confirm several different centroids is what we need to do first, and the number of the centroids is our first level's k . Then during all these small data group(subset), we can find some of them share some common trait, and this trait is defined by us. We call the new subset is level 2. Then we can keep this work go on and go on, until we arrive the root.

3.2 Multilevel techniques

Multilevel models are known by several names: hierarchical models, generalized linear mixed models, nested models, and split-plot designs [4]. Because in our project we introduce a combination of the multilevel paradigm with a popular algorithm used to solve the clustering problem. So our finally work's structure will like a tree map.

At the top is our object data-base, and at the lowest level our object been divided into several different small part.

So we know both K-clustering and multilevel techniques. What can we do now? We know that each data object has its own factors, and in the course of cluster them we find that we can use knapsack problem's solution in our work. And now we will explain what are knapsack problem, and its solution we use in our project and why we chose.

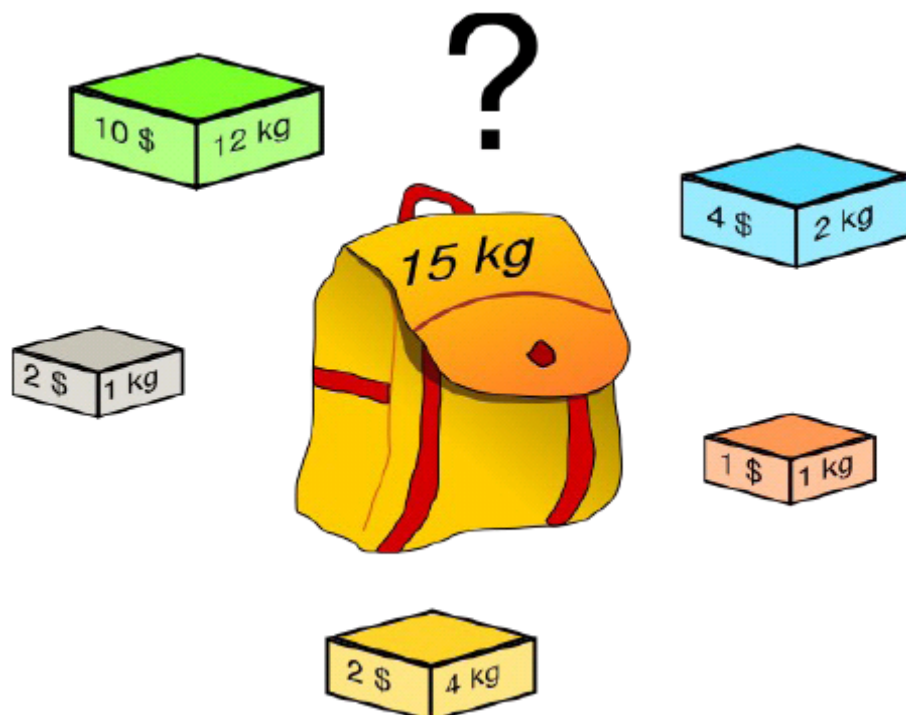


Figure 1 the Knapsack problem

The knapsack problem can be described like this: a thief goes into a museum carrying a knapsack with limited capacity, and he wants to steal as much of value as possible without his knapsack being exceeded, this means his solution is to let the item put in his knapsack has the biggest value.

As we said we have two common types of the knapsack problem-fractional knapsack and binary knapsack. In fractional knapsack the thief can put any percentage of the items into the knapsack, so we can see that in this kind of knapsack our solution is take as much as possible of the most valuable item, when this item is empty we keep doing this in the second item, and we will continue to do this until the knapsack is full. But the binary knapsack is totally different; in binary knapsack the thief can put only 100% or 0% of the item into the knapsack. Through our calculation we know that fractional knapsack is much more useful than binary knapsack to our problem when

we choose the data from our objects. But in reality when we download a webpage we download the whole page or none of it, so this means when we evaluate a webpage's value we can use fractional knapsack, but when we download a webpage the choice we have is download 100% or 0%, so this is very familiar with binary knapsack.

3.3 Stochastic

Before our work, there is a lot of people try to find an easy way that can automatic evaluate a webpage's value. The most popular one is stochastic. Now we will first introduce what is stochastic.

Stochastic is synonymous with "random." The word is of Greek origin and means "pertaining to chance" [5]. Therefore, stochastic models are based on random trials.

Because the time of a webpage been visited is random so we can see it as a stochastic model and try to find an automatic way for our computer or searching engine to evaluate its value. In real application people combine stochastic model with nonlinear equality fractional knapsack problem (NEFKP) to deal with such problem.

In this dealing method, each web page is connected to a learning automata, each learning automata has a defined probability of being visited. When a page is downloaded and it is uploaded, the probability increases, but when this page is downloaded and not updated, the probability will decrease. This probability we can use pagerank to describe it, through pagerank we can know which page is most popular, and we can speculate that the most popular page must be much more valuable than any others. So if a pagerank is 0.5, next we will look for the webpage whose pagerank is higher than 0.5. We keep doing this all the way until we find no higher pagerank than the last we found, and then we find the best.

In searching engine this is why they make a list of all the webpage that has the similar information.

4. Problem description

In this project we combine K-clustering and multilevel to deal such problems. We will first introduce how is our design work and later we will compare the difference between this them.

In our combination work we use 1000 data as our object, first we cluster them. In this course we needn't to care about it is familiar with each other or not. We just need to go directly divide it. Like 1000 part to 500, 500 divided to 250. This work keep going until we go to the lowest level, the subset's element should be less (or equal) than 10. Then we go to next stage, we use fractional knapsack to find the best data in each subset and drop the rest data in this subset, this will be done in each subset. After this we take these data to higher level and in there we will don the same job. This work will keep going until we arrive the highest point which is our best answer. And the rest of them will be dropped in this course. And all data in the final result is the best answer in this job.

Now we can see the difference between our work and stochastic. Our combination work is in a static environment, but stochastic is obviously dealing the dynamic situation; and another difference is stochastic is dealing the objects that is nonlinear.

5. Requirement specification for K-clustering & multilevel techniques

As we mentioned, we will use the solution of knapsack problem in our clustering problem. In two Knapsack problem solutions-binary and fractional, because in every web page there it has a lot of information on it, when we can not say this webpage is not good enough when it has some useless data. So since this we need to consider it all-sided. And because of this, when we evaluate the value of a webpage we use fractional knapsack.

In the multilevel process we can not divide the data object all the way until each subset has only one data, this for us is insignificance. So as we see we need to decide how many levels in our multilevel structure and how big of our lowest subset.

We already know how our design work is. And next we need to know the efficiency of it, and does it really much better than people did before? We will give our evaluate result and compare later.

In this whole testing and evaluation course large random data sets will be generated in local computer in order to evaluate the quality of the clustering.

6. Design of combination of K-clustering and multilevel techniques

In this project we are mainly make a combination of multilevel technique and K-clustering. The main question is how to find the most valuable objects in a huge cluster of objects. So the Knapsack problem is the most important problem we should solve first.

6.1 List of Knapsack problems

The general way to view a knapsack problem is that of a bag of limited capacity, which is to be filled while maximizing the value of the objects in it. [6]

As the explanation from USACO, the knapsack problem style has three forms:

Fractional knapsack problem

A fractional knapsack problem is one in which you are allowed to place fractional objects in the knapsack. This is the easiest form of the knapsack problem to solve.

Integer Knapsack problem

In integer knapsack problems, only complete objects can be inserted into the knapsack.

Multiple knapsack problem

In the multiple knapsack problem, more than one knapsack is to be filled. If fractional objects are allowed, this is the same as having one large knapsack with capacity equal to the sum of all the available knapsacks, so this term will only be used to refer to the case of multiple integer knapsacks.

Obviously, the fractional knapsack problem is the easiest of the three to solve. But when we got a lot of data, or we open a web page, it's impossible to evaluate them by fractional. So our problem is focus on integer Knapsack problem. Some also called it Binary Knapsack problem.

6.2 algorithms of Knapsack problem

There are hundred of algorithms which can implement Knapsack problem.

6.2.1 Value Density (Greedy solution)

“Find the object which has the highest “value density” (value of weight/value). If the total amount of capacity remaining exceeds the availability of that object, put all of it in the knapsack and iterate. If the amount of capacity is less than the availability of the object, we can use as much as possible and terminate. This algorithm runs in $N \log N$ since it must sort the objects first based on value density and then put them into the knapsack in decreasing order until the capacity is used. It's normally easier to not sort them but rather just keep finding the highest value density not used each time, which gives an $O(N^2)$ algorithm.” [6]

6.2.2 Dynamic Programming

Do dynamic programming on the maximum value that a knapsack of each size can have in it. A simple one is to have an algorithm that runs a fixed number of iterations (100000 iterations). When we do each time of iteration, we choose two clusters C_i and C_j at random. C_i is the knapsack and C_j not in the knapsack. Exchange the objects in these two clusters and see if the profit gets better while the weight of knapsack is repeated. If these two conditions are satisfied, then we can update our solution. After 100000 iterations, it should be the best solution of Knapsack problem.

6.2.3 Optimal Binary Search Trees

We use depth priority strategy in return method to searching a solution in a tree space which is formed in a tree structure.

When we search solution by this method in the tree structure, whenever we arrive at a node, we always to judge is this node contains our solution. If it is not we will ignore the entire rest node which is use this node as its root, and go back to its father node. But if it contains the solution we need, we will go to the next son-node, keep doing the same work; and this node we will call it extendable node. When we use return method to search all the solution we need, we need to go back to the root, and search all son tree which use the entire nodes as its root; when we can not find any new extension point then we can stop our searching work. If we just looking for one answer then we just need to search find one and our searching work can stop. This is how return method works, it can be used in those contains very big data-base. Translate from [7]

6.3 Multilevel strategy

The data in database or Web resource is numerous. Data mining is the process of automatically searching large volumes of data. It applies many computational techniques such as statistics, machine learning and pattern recognition. But when we compute those data directly, our storage of computer is limited, and the speed of computing the data will be really slow. So we can use a strategy of multilevel cluster the data. And it can be easily used more than one computer. Also it increases the speed of computing a lot.

As we thought we will first divide the entire element in our object data-base into several different groups, this we considered as our first level, and the second level is to group all these 1st level's group into a bigger group, this is the second level. We keep this work going, and finally we will reach the root. So we can get a tree map when our work is finished.

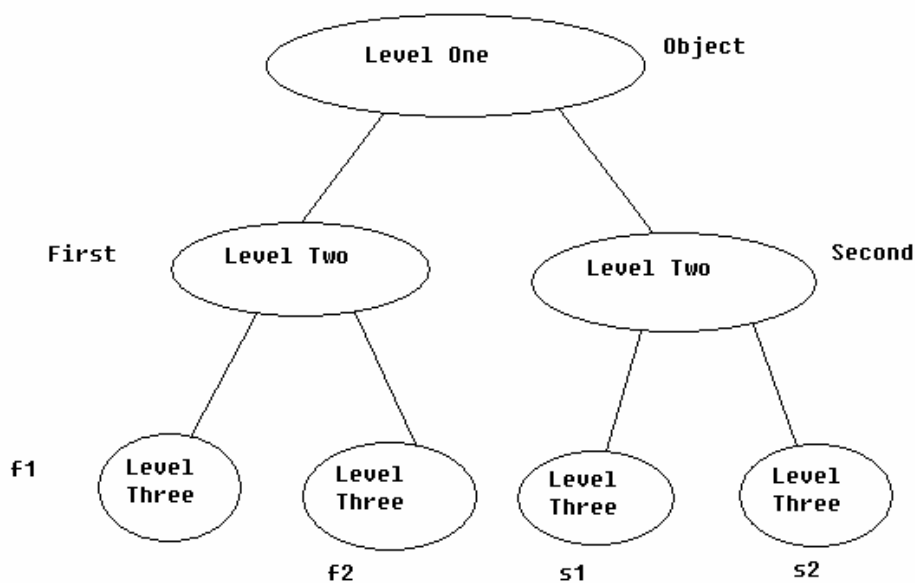


Figure 2 Structure of multilevel strategy

As the picture above, we divide the data by three levels for instance. The Level One is the first level, and I named the cluster as Object. Then I divide it into two clusters. That's the Level Two. And the left cluster named First, the other one is Second. Let's continue to the Level Three. Each cluster of Level Two has divide into two clusters again. Consider for convenience, they called f1, f2, s1, and s2.

When we do the multilevel strategy, first we calculate the knapsack solution of f1 and f2. Then we drop the objects which not inside the knapsack, and add both as a new cluster First. So do the clusters of s1 and s2. After calculate the solution of the

newFirst and newSecond. We add them again as a new Object. So we have fewer objects in the biggest cluster Object. Then we calculate it again and get the final solution of those data.

6.4 flow chart of program

Here is the flow chart of the program. We have two classes. One is for the whole objects in the cluster. And we have several types of data in the class of MyObject. They represent the value, weight and the rate of value and weight. There is another class which contains the main program in it named Knapsack. You can clearly see that we have several methods in it which we defined it ourselves. We can learn the purpose of method easily by their names. As divide for example, it is used when we need to divide the cluster into two or more than two clusters. The process of the program is very obviously. We used the data by random.

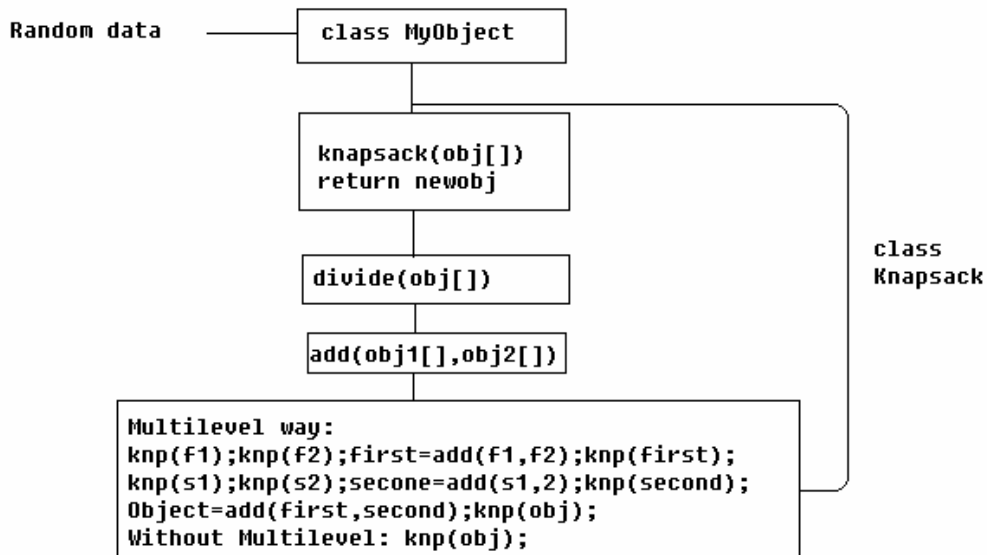


Figure 3 the flow of program

The following picture is an example which can show a blue print of our work, and this picture can visualize our work to you.

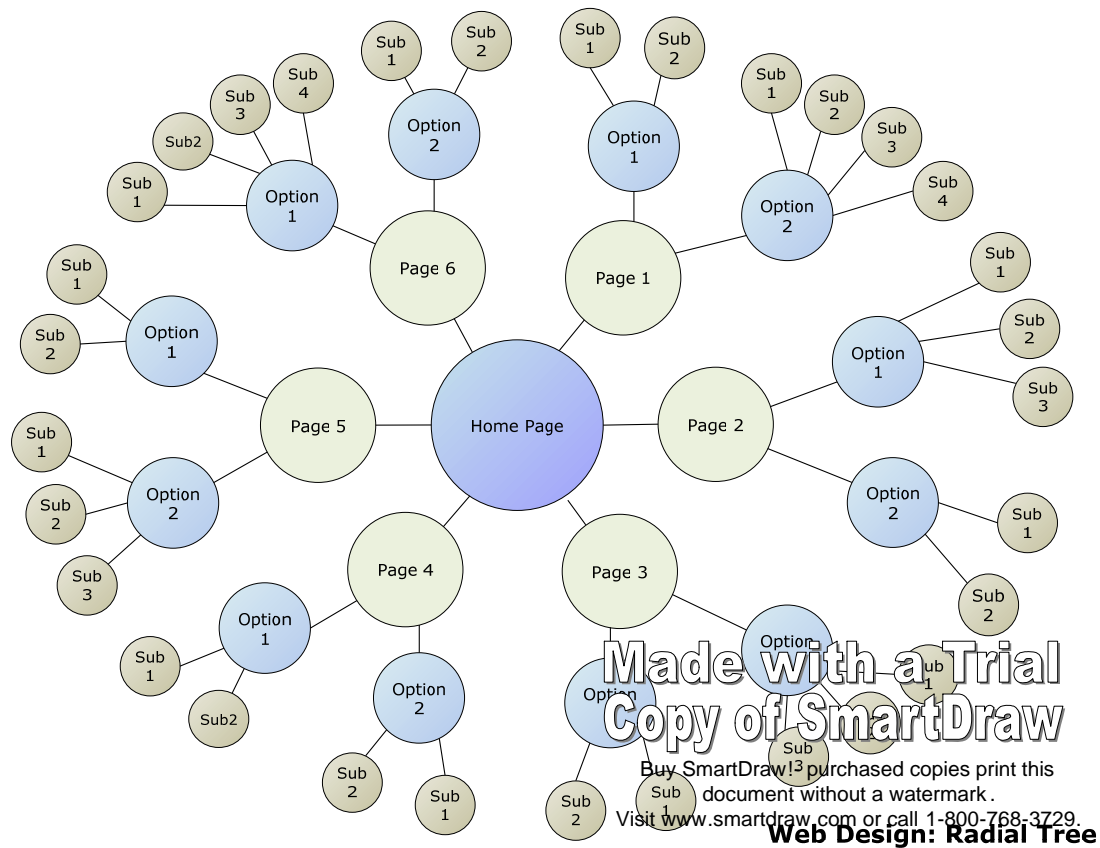


Figure 4 blue prints

In this tree map we use a home page (the blue center) as our object, and try to cluster and multilevel its data-base. In this progress we first cluster this page’s content into several different pages (in this example we have 6 pages in grey), and each page’s data is similar, and all of them are in the same level. Our work can keep going to next level which showed in light blue in this tree map. When this progress finished, we can reach the subset level which is the lowest level in this progress. Now we have a multi level structure, and this is exactly what we imagine.

Now we get this map, but can we stop now. Of course not, we still need to use fractional knapsack to choose the best data-base, then we take these data to the higher level. In this level we do the same procedural. Finally when we arriving root we can get the best data for our problem.

In our work, we first define an algorithm of our clustering and program it by Java.

7. Implementation

7.1 Failed with Dynamic programming

When we cluster our data by multilevel way, for example our input data (file = 1000 objects), then we use the multilevel strategy until we get 5% for instance of the initial number of objects. 5% of 1000 is 50. Now our problem has 50 clusters. Each cluster has several objects in it. Then we try to find an initial solution to the knapsack problem that maximizes the profit while not violating the constraint of weight to improve the solution. If we don't find a solution, then go back from 50 to 100 and try to find an initial solution. Once an initial solution is obtained, we need to improve the solution. We can several strategies. The first strategy we considered is dynamic programming. It's the simplest one to have an algorithm that runs a fixed number of iterations (100000 iterations). Update this array for an object of size S by traversing the array in reverse order (of capacity), and seeing if placing the current object into the knapsack of size K yields a better set than the current best knapsack of size $K+S$. This algorithm runs in $K \times N$ time, where K is the size of the knapsack, and N is the sum of availability of objects. If the knapsack is too large to allocate this array, recursive descent is the method of choice, as this problem is NP-complete. Of course, recursive descent can run for a very long time in a large knapsack being filled with small objects. After $K \times N$ times iterations, we go to next level, from 50 to 100 clusters and do the improvement algorithm once again. This continues from until we get the initial problem of 1000 objects.

It's not a very simple strategy when we implement it in programming. So we change the strategy to implement the Knapsack problem. We failed to implement it by exchange the data $K \times N$ times, and we didn't check the information about times of iterations, and make mistake about the meaning of 100000 iterations. At last we decide to change the algorithm instead of greedy solution.

I try to use an array represent the value of the objects and another array represent the weight of the objects. And I can use a boolean to represent whether the object in the knapsack or not.

7.2 Implement in Greedy algorithm

So we come to cluster the objects (1000) by greedy algorithm in multilevel way. I still get 5% that's 50 clusters of objects. As the same, I can use an array represent the

weight of cluster of objects' weight. And each weight is the sum of the entire object's weight in the cluster. So do the value. Then I can calculate the rate of value and weight as value/weight. And sort them from the highest to lowest. And select them from highest to lowest and test if it's fit the constraint of the knapsack every time I select one. Then I can get the maximal value of the knapsack problem. Because we have to multilevel cluster the data. So we compute the knapsack problem by greedy algorithm in the bottom level first. After we get the solution of each cluster, we drop the objects which are not inside the knapsack problem and add them back to the next level from bottom. And keep it go on, until the top objects ordered finally.

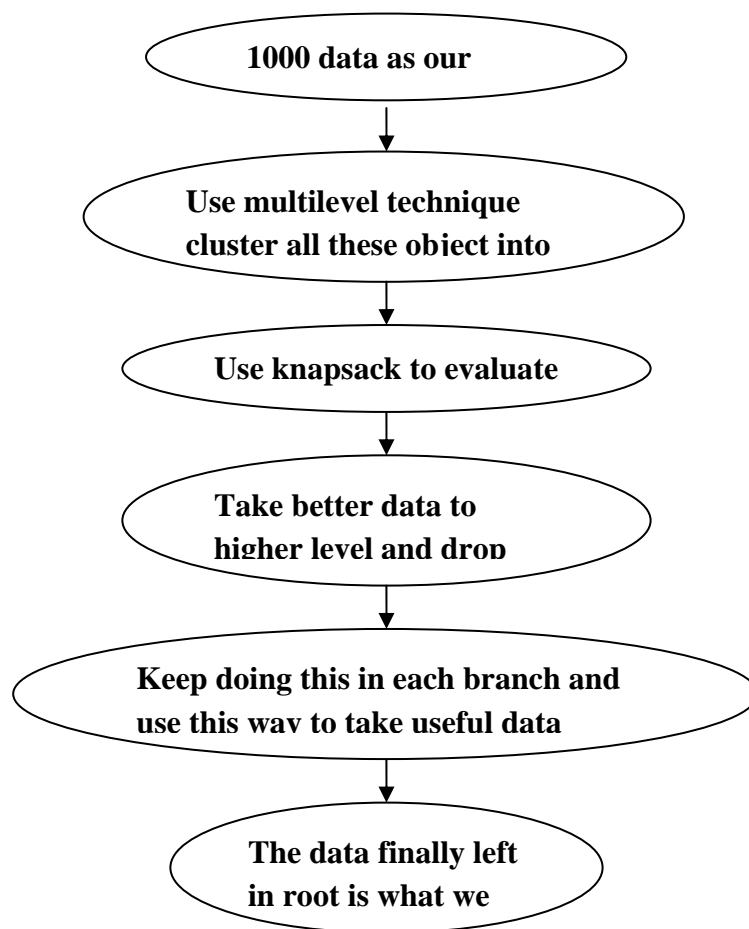


Figure 5 procedure

The main coding are:

```
Class MyObject {float v, w, p;  
                boolean r=false;  
};
```

```

Class Knapsack{
    public class knp(obj)
    { ...
        return newobj;
    } //to calculate the solution of knapsack problem in object and get the new object.

    public class divide(obj)
    { ...
        return newobj[obj1][obj2];
    } //divide one cluster into two cluster;

    public class add(obj1,obj2)
    { ...
        newobj=[obj1+obj2];
        return newobj;
    } //add two cluster into one cluster;

    public static void main(String args)
    { ...
        MyObject obj[] = new MyObject[size];
        divide(obj); //creat the second level;
        divide(obj1); divide(obj2); // creat the third level;
        knp(f1); knp(f2); // get the new objects of the third level;
        obj1=add (f1, f2); knp (obj1);
    // get the new objects of the first cluster in level two;
        knp(s1); knp(s2); obj2= add(s1,s2); knp(obj2);
    // do the same with the other cluster in level two;
        Object = add(obj1,obj2); knp(obj);
    // finish the multilevel clustering
    ... }
}

```

8. Evaluation and testing

8.1 Coding

8.1.1 Testing

I had thought a lot about the algorithm we choose in coding. At first, I chose the dynamic programming strategy. It's the optimal solution of integer Knapsack problem, and recommended by our supervisor. But we can not find out the algorithm of calculating the times of iterations. We were very confused about how to explain 100000 iterations in coding. Then we decided the simplest algorithm, Greedy algorithm. After we finish the coding part of Greedy algorithm, we found out the optimal binary search trees can work out too. And its structure is clearer than the Greedy algorithm. And it can combine to the multilevel strategy structure which we will talk about next. But because of the time and energy, we couldn't go through the coding part of the binary trees. And the result we implement in Greedy algorithm is very good.

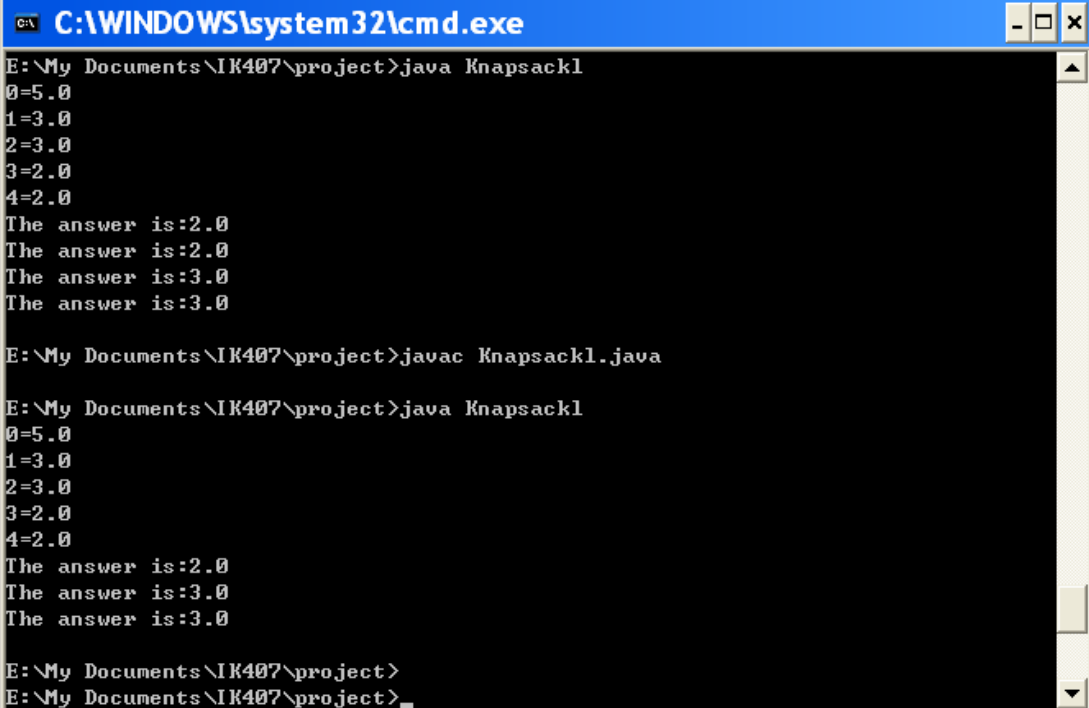
8.1.2 Evaluation

Evaluating those three algorithms, the Greedy solution is the simplest one, and dynamic programming algorithm is slightly complex than greedy solution. Of course, the Optimal Binary Search Trees comes the most complex one to implement. We take the greedy solution as an example of coding. Fractional values are not a problem; the array just becomes an array of real numbers instead of integers. Fractional availability doesn't affect things, as you can, without loss of generality, truncate the number (if you have 3.5 objects, you can only use 3). Fractional size is a pain, as it makes the problem recursive descent. If the sizes are all the same, the problem can be solved greedily, picking the objects in decreasing value order until the knapsack is full. If the values are all 1.0, then again greedy works, selecting the objects in increasing size order until the knapsack is full. [7]

8.2 results

8.2.1 The first testing

After finished the coding part, we had do some test to testing the program. This is a result of the initial program we made for the first time that can implement that basic problem.



```
C:\WINDOWS\system32\cmd.exe
E:\My Documents\IK407\project>java Knapsack1
0=5.0
1=3.0
2=3.0
3=2.0
4=2.0
The answer is:2.0
The answer is:2.0
The answer is:3.0
The answer is:3.0

E:\My Documents\IK407\project>javac Knapsack1.java

E:\My Documents\IK407\project>java Knapsack1
0=5.0
1=3.0
2=3.0
3=2.0
4=2.0
The answer is:2.0
The answer is:3.0
The answer is:3.0

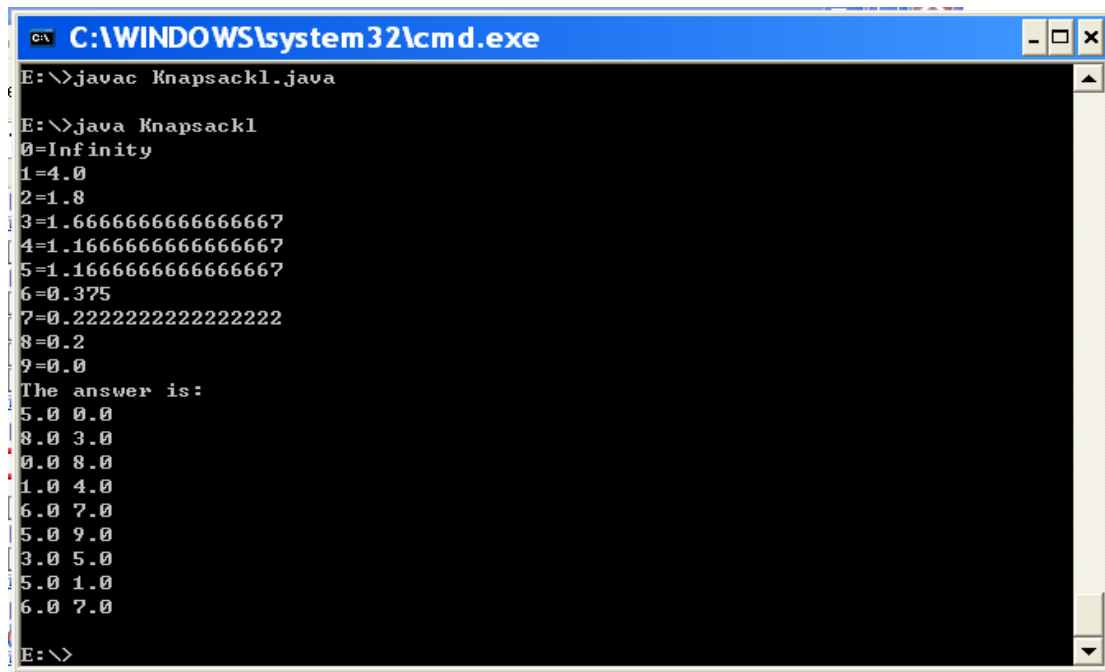
E:\My Documents\IK407\project>
E:\My Documents\IK407\project>
```

Picture 6 The first result

We test the first program by 5 objects, and we used the type of the data is int. The first five numbers represent the rate of value and weight of each object. And the answer is the solution of the Knapsack solution of the 5 objects. The numbers is the value of the value divide weight of the objects in the knapsack.

8.2.2 The second testing

The first program is testing by the int numbers directly in the program. We create an abstract class that can easy make examples when we call the method in program. And test it by 10 int data again.



```

C:\WINDOWS\system32\cmd.exe
E:\>javac Knapsack1.java
E:\>java Knapsack1
0=Infinity
1=4.0
2=1.8
3=1.6666666666666667
4=1.1666666666666667
5=1.1666666666666667
6=0.375
7=0.2222222222222222
8=0.2
9=0.0
The answer is:
5.0 0.0
8.0 3.0
0.0 8.0
1.0 4.0
6.0 7.0
5.0 9.0
3.0 5.0
5.0 1.0
6.0 7.0
E:\>

```

Figure 7 second testing

The first row of numbers stills the value of value divide weight of testing objects. But we change to display the value and the weight of each object which is the solution, instead of just the value of weight divided by value.

8.2.3 Third

But when we come to 1000 objects, it can not show the result. And it shows out of range. So we improve the program as enlarge the range of each data type. Now we can display the result of 1000 objects.

```

C:\WINDOWS\system32\cmd.exe
6.0 0.0
6.0 5.0

E:\>javac Knapsack1.java

E:\>java Knapsack1
The weight of answer is:
3.0 5.0 1.0 4.0 2.0 7.0 8.0 2.0 3.0 1.0 2.0 0.0 3.0 5.0 0.0 6.0 3.0 0.0 2.0 8.0
6.0 2.0 8.0 2.0 8.0 8.0 0.0 0.0
E:\>javac Knapsack1.java

E:\>java Knapsack1
We have : 1000 objects!
The weight of answer is:
5.0 2.0 3.0 4.0 2.0 7.0 5.0 1.0 3.0 1.0 4.0 1.0 6.0 7.0 0.0 2.0 0.0 5.0 9.0 5.0
3.0 1.0 2.0 0.0 8.0 2.0 9.0 1.0 0.0 3.0 The value of answer is:

E:\>javac Knapsack1.java

E:\>java Knapsack1
We have : 1000 objects! The constraint of knapsack is: 100.0
The value of answer is:
0.0 8.0 7.0 9.0 2.0 3.0 5.0 5.0 8.0 4.0 4.0 0.0 7.0 9.0 7.0 4.0 1.0 1.0 4.0 4.0
9.0 2.0 4.0 7.0 8.0 6.0
E:\>

```

Figure 8 third testing

It shows very clear that the numbers of testing objects, the constraint of knapsack, and the value of the object of the solution. But it just shows the int data that can implement clearly. We can't decide the type of data from web source or in the database, so we should enlarge the range of data again.

8.2.4 Fourth

We add method of getting data by random. And the type of the data should enlarge to double. Here are the results of 10 objects testing by random and 1000 objects also.

```

C:\WINDOWS\system32\cmd.exe
E:\My Documents\IK407\project>javac Knapsack1.java

E:\My Documents\IK407\project>java Knapsack1
0=1.7916218466234584
1=1.5322878122664814
2=1.5101706009969653
3=1.2766406214501318
4=1.1331614608476115
5=1.003331598531019
6=0.19170317308243356
7=0.15529713753219423
8=0.09277256450758219
9=0.08793031410058975
The answer is:
4.514424613353514 0.7010772200586834
0.4745309187732627 0.7166226427954603
9.198582586651474 0.853376096398446
6.319265010010099 6.340318264034568
4.223028573955926 7.566070252014545
2.371637523791045 3.6340312728188753
8.342633148514395 0.7335703531748627
9.791513422928958 1.8770641924547216
7.358728449028974 8.33862747928255

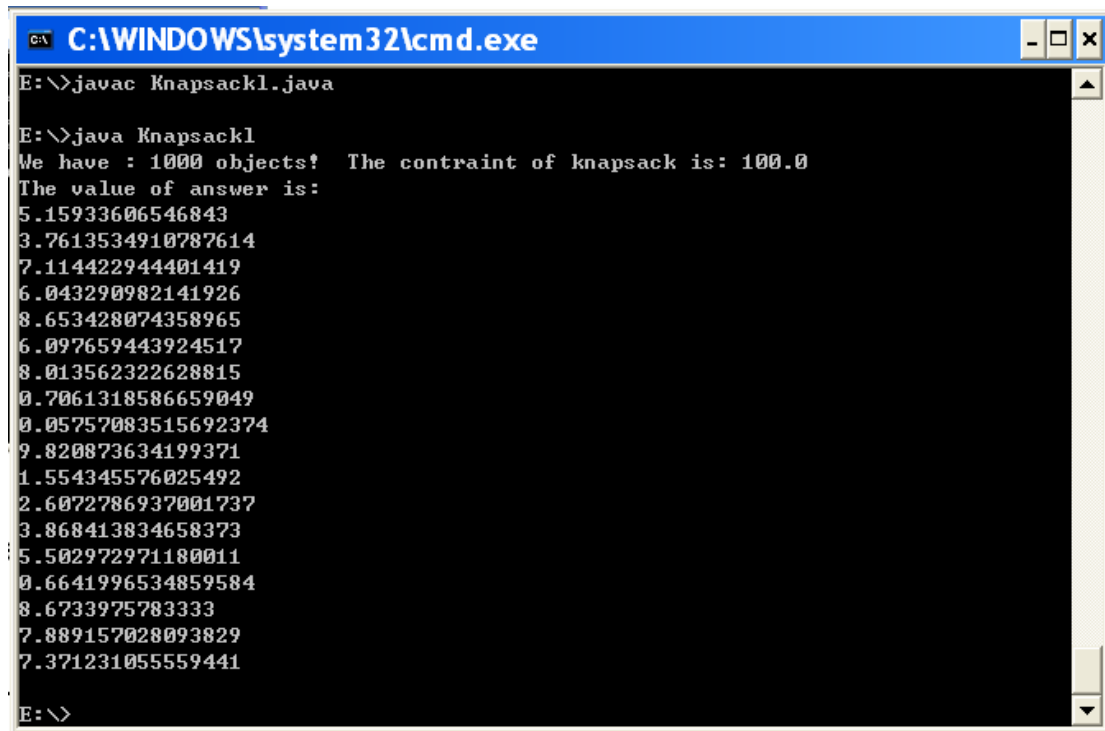
E:\My Documents\IK407\project>

```

Figure 9 fourth

What a mess of numbers! Don't worry. It is ordered. The first 10 number still the value of value/weight. And the several numbers below are the value of weight and the value of each object of solution. The left raw is representing the value of weight, while the right raw is the value of value.

I made the next testing more clear to show the solution. Because its 1000 objects testing.



```
C:\WINDOWS\system32\cmd.exe
E:\>javac Knapsack1.java
E:\>java Knapsack1
We have : 1000 objects! The constraint of knapsack is: 100.0
The value of answer is:
5.15933606546843
3.7613534910787614
7.114422944401419
6.043290982141926
8.653428074358965
6.097659443924517
8.013562322628815
0.7061318586659049
0.05757083515692374
9.820873634199371
1.554345576025492
2.6072786937001737
3.868413834658373
5.502972971180011
0.6641996534859584
8.6733975783333
7.889157028093829
7.371231055559441
E:\>
```

Figure 10 fourth2

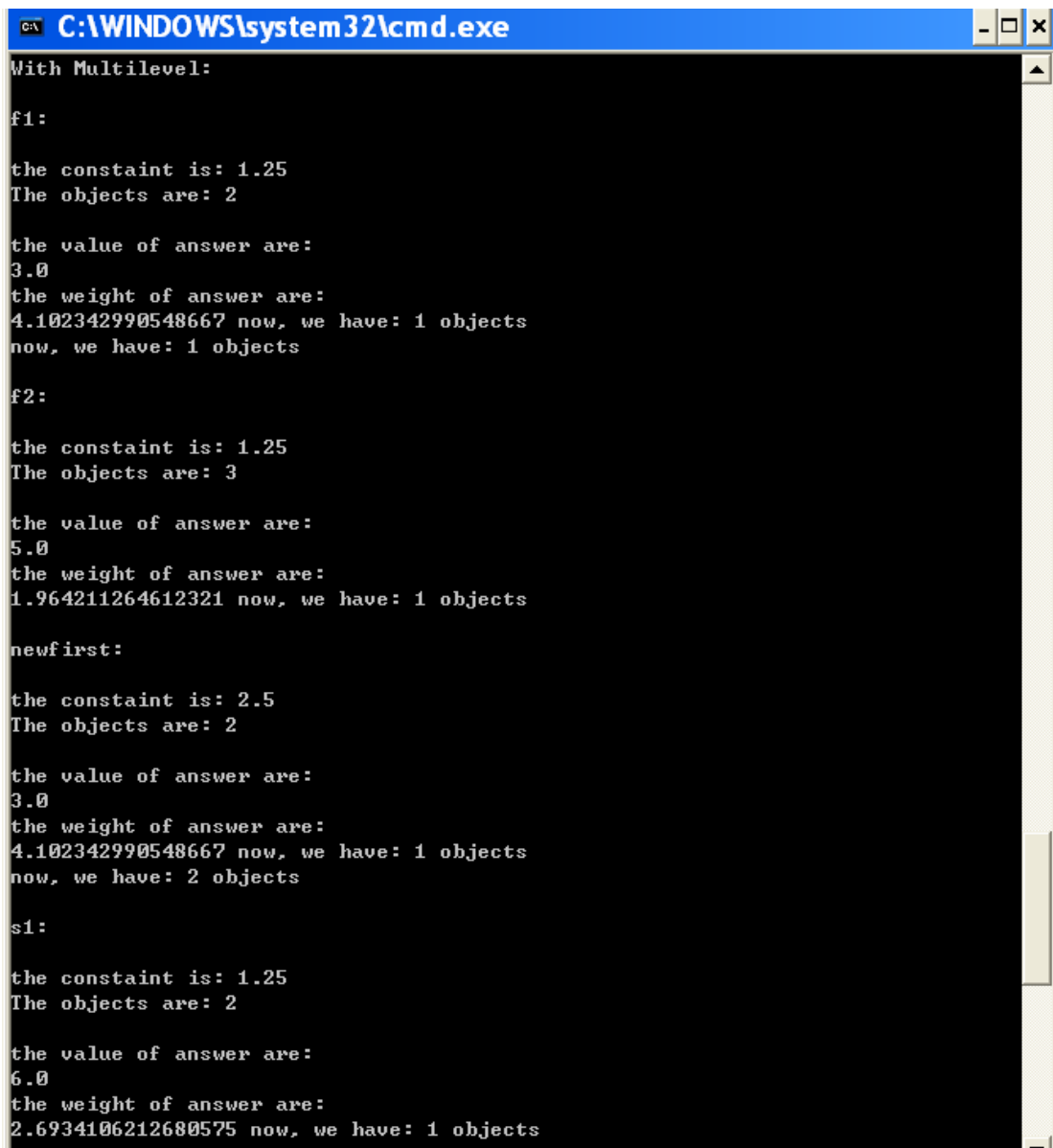
As showing above, we can easily learn the meaning of numbers from the outcome.

8.2.5 Final testing

The tests above just are the test of implementation of Knapsack problem. The big problem is to multilevel clustering the data.

We still have two tests. One is 10 objects. And the other is 1000 objects.

10 objects



```
C:\WINDOWS\system32\cmd.exe
With Multilevel:

f1:

the constaint is: 1.25
The objects are: 2

the value of answer are:
3.0
the weight of answer are:
4.102342990548667 now, we have: 1 objects
now, we have: 1 objects

f2:

the constaint is: 1.25
The objects are: 3

the value of answer are:
5.0
the weight of answer are:
1.964211264612321 now, we have: 1 objects

newfirst:

the constaint is: 2.5
The objects are: 2

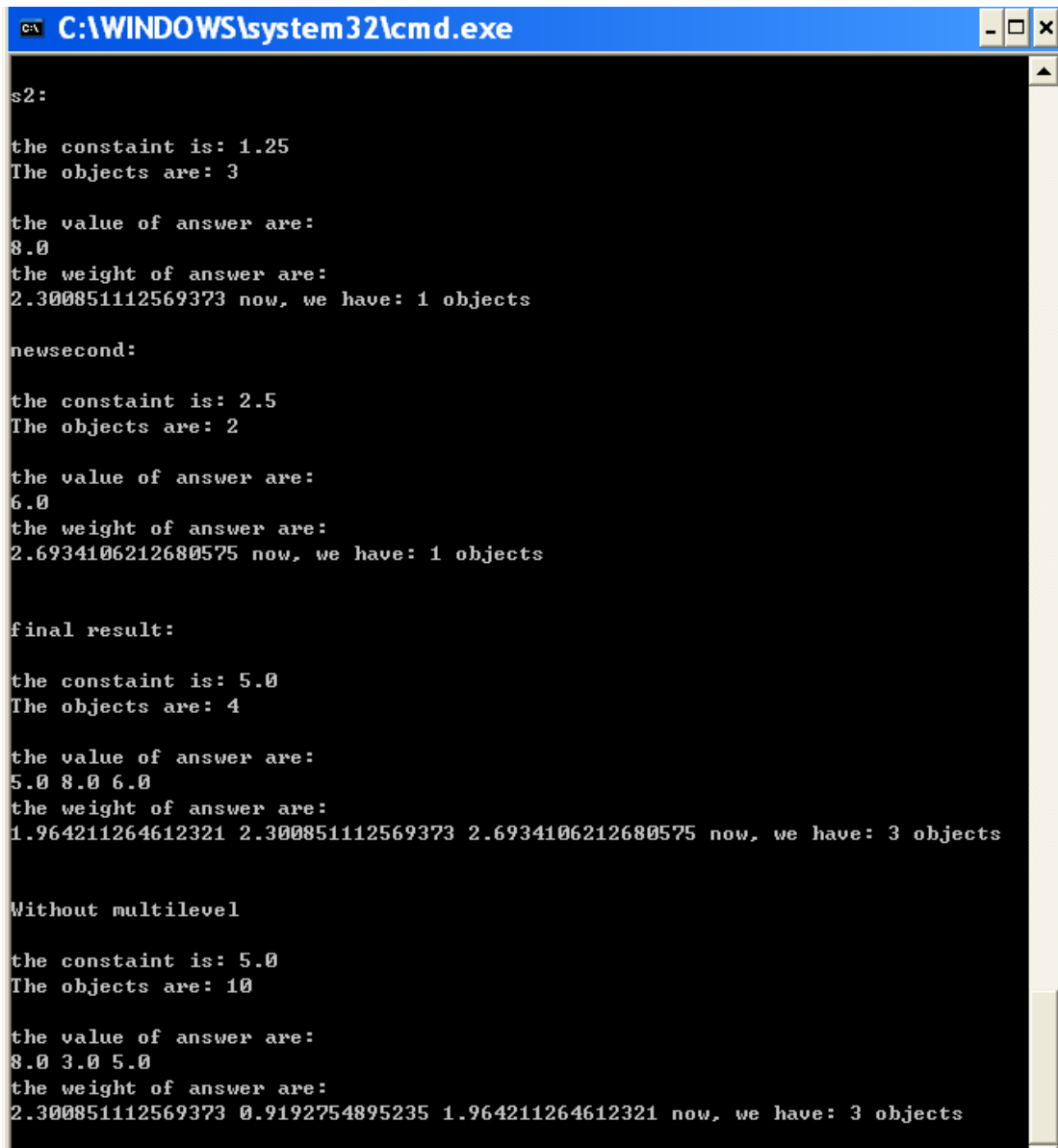
the value of answer are:
3.0
the weight of answer are:
4.102342990548667 now, we have: 1 objects
now, we have: 2 objects

s1:

the constaint is: 1.25
The objects are: 2

the value of answer are:
6.0
the weight of answer are:
2.6934106212680575 now, we have: 1 objects
```

Figure 11 final testing1



```

C:\WINDOWS\system32\cmd.exe
s2:
the constaint is: 1.25
The objects are: 3

the value of answer are:
8.0
the weight of answer are:
2.300851112569373 now, we have: 1 objects

newsecond:
the constaint is: 2.5
The objects are: 2

the value of answer are:
6.0
the weight of answer are:
2.6934106212680575 now, we have: 1 objects

final result:
the constaint is: 5.0
The objects are: 4

the value of answer are:
5.0 8.0 6.0
the weight of answer are:
1.964211264612321 2.300851112569373 2.6934106212680575 now, we have: 3 objects

Without multilevel
the constaint is: 5.0
The objects are: 10

the value of answer are:
8.0 3.0 5.0
the weight of answer are:
2.300851112569373 0.9192754895235 1.964211264612321 now, we have: 3 objects

```

Figure 12 final testing2

1000 objects

We showed each objects of cluster from the bottom level to the top level. The number of objects is huge. So the screen shots are a lot. But we can easily see the difference between them.

The first picture shows us how we divide the first step of multilevel works.

```

C:\WINDOWS\system32\cmd.exe
f1:
the constaint is: 12.5
The objects are: 250

the value of answer are:
62.0 47.0 68.0 91.0 49.0 48.0 94.0 89.0
the weight of answer are:
0.5102091999883429 0.590891721841591 1.5396543318352718 2.07755075256415 1.25949
12886241816 1.4939672365767875 3.3948320308852864 3.8566927851949706 now, we hav
e: 8 objects
now, we have: 8 objects

f2:

the constaint is: 12.5
The objects are: 250

the value of answer are:
25.0 62.0 99.0 73.0 84.0
the weight of answer are:
0.6462444518469113 1.7417345440543186 3.105654578895478 5.169765625470069 7.3482
25538557651 now, we have: 5 objects

newfirst:

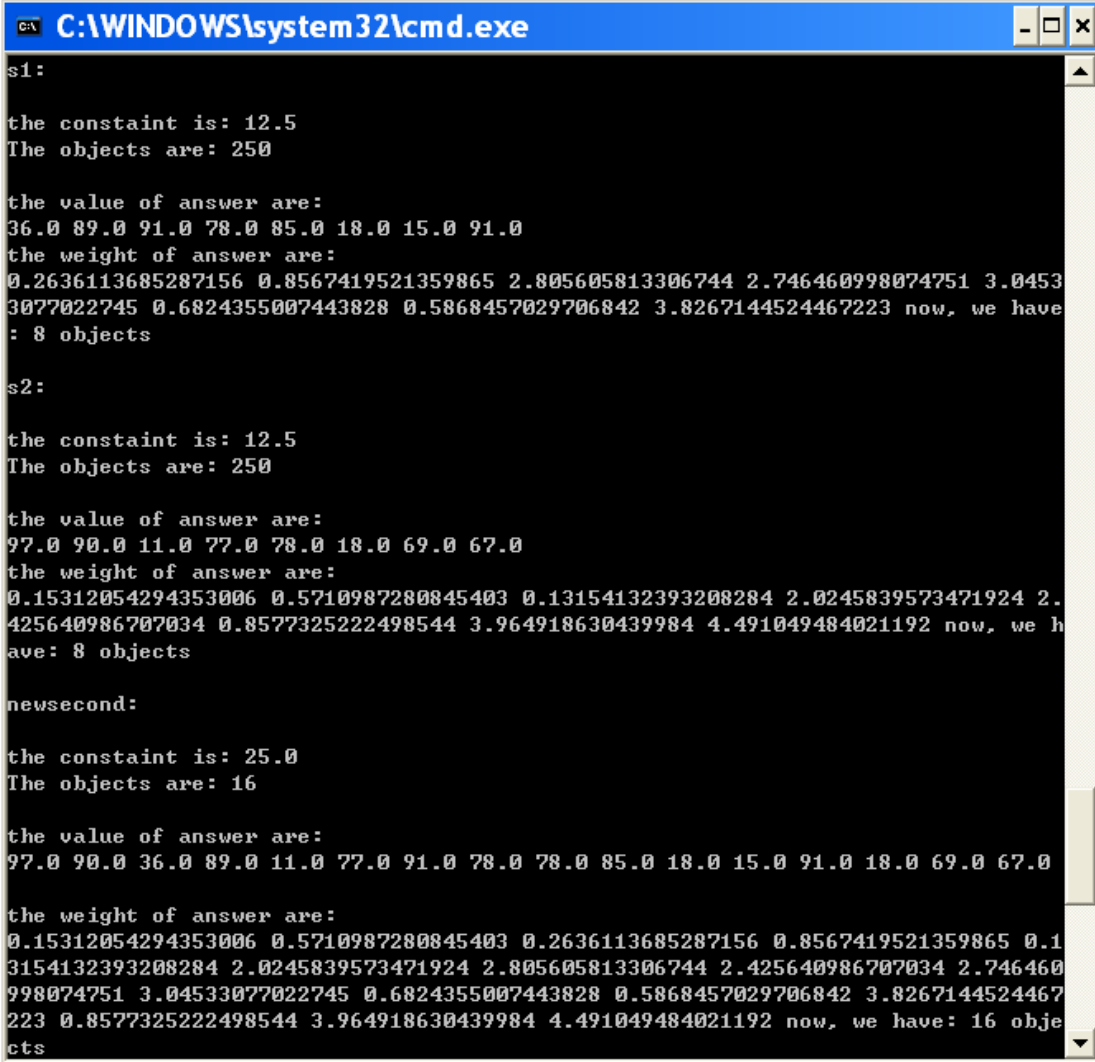
the constaint is: 25.0
The objects are: 13

the value of answer are:
62.0 47.0 68.0 91.0 49.0 25.0 62.0 48.0 99.0 94.0 89.0 73.0
the weight of answer are:
0.5102091999883429 0.590891721841591 1.5396543318352718 2.07755075256415 1.25949
12886241816 0.6462444518469113 1.7417345440543186 1.4939672365767875 3.105654578
895478 3.3948320308852864 3.8566927851949706 5.169765625470069 now, we have: 12
objects
now, we have: 13 objects

```

Figure 13 final testing3

f1, f2 are the cluster of the bottom level. They combined again after get the solution of each cluster and become a new First cluster of level two. We can see the numbers of cluster after they get the solution of knapsack, and drop the objects useless. And the constraint changes when it used in different level.



```

C:\WINDOWS\system32\cmd.exe
s1:
the constaint is: 12.5
The objects are: 250

the value of answer are:
36.0 89.0 91.0 78.0 85.0 18.0 15.0 91.0
the weight of answer are:
0.2636113685287156 0.8567419521359865 2.805605813306744 2.746460998074751 3.0453
3077022745 0.6824355007443828 0.5868457029706842 3.8267144524467223 now, we have
: 8 objects

s2:

the constaint is: 12.5
The objects are: 250

the value of answer are:
97.0 90.0 11.0 77.0 78.0 18.0 69.0 67.0
the weight of answer are:
0.15312054294353006 0.5710987280845403 0.13154132393208284 2.0245839573471924 2.
425640986707034 0.8577325222498544 3.964918630439984 4.491049484021192 now, we h
ave: 8 objects

newsecond:

the constaint is: 25.0
The objects are: 16

the value of answer are:
97.0 90.0 36.0 89.0 11.0 77.0 91.0 78.0 78.0 85.0 18.0 15.0 91.0 18.0 69.0 67.0

the weight of answer are:
0.15312054294353006 0.5710987280845403 0.2636113685287156 0.8567419521359865 0.1
3154132393208284 2.0245839573471924 2.805605813306744 2.425640986707034 2.746460
998074751 3.04533077022745 0.6824355007443828 0.5868457029706842 3.8267144524467
223 0.8577325222498544 3.964918630439984 4.491049484021192 now, we have: 16 obje
cts

```

Figure 14 final testing4

This also shows the process of multilevel cluster. But it is the other left clusters calculating from bottom level to the second level.

The last figure is the solution of multilevel and without multilevel.

We can compare them easily. The number of objects in multilevel is obviously less than the solution without multilevel. Multilevel clustering get 29 objects in the final cluster while clustering without multilevel still get 1000 objects.

And we found an interesting discovery that the final solution with multilevel clustering is different from the one without multilevel. We will discuss it in the discussion part.

```

C:\WINDOWS\system32\cmd.exe

final result:

the constaint is: 50.0
The objects are: 29

the value of answer are:
97.0 90.0 36.0 62.0 89.0 11.0 47.0 68.0 91.0 49.0 25.0 77.0 62.0 91.0 78.0 48.0
99.0 78.0 85.0 94.0 18.0 15.0 91.0 89.0 18.0 69.0 67.0 73.0
the weight of answer are:
0.15312054294353006 0.5710987280845403 0.2636113685287156 0.5102091999883429 0.8
567419521359865 0.13154132393208284 0.590891721841591 1.5396543318352718 2.07755
075256415 1.2594912886241816 0.6462444518469113 2.0245839573471924 1.74173454405
43186 2.805605813306744 2.425640986707034 1.4939672365767875 3.105654578895478 2
.746460998074751 3.04533077022745 3.3948320308852864 0.6824355007443828 0.586845
7029706842 3.8267144524467223 3.8566927851949706 0.8577325222498544 3.9649186304
39984 4.491049484021192 5.169765625470069 now, we have: 28 objects

Without multilevel

the constaint is: 50.0
The objects are: 1000

the value of answer are:
97.0 90.0 36.0 62.0 89.0 11.0 47.0 68.0 91.0 49.0 25.0 77.0 62.0 91.0 78.0 48.0
99.0 78.0 85.0 94.0 18.0 15.0 91.0 89.0 47.0 23.0 85.0 18.0 18.0 83.0
the weight of answer are:
0.15312054294353006 0.5710987280845403 0.2636113685287156 0.5102091999883429 0.8
567419521359865 0.13154132393208284 0.590891721841591 1.5396543318352718 2.07755
075256415 1.2594912886241816 0.6462444518469113 2.0245839573471924 1.74173454405
43186 2.805605813306744 2.425640986707034 1.4939672365767875 3.105654578895478 2
.746460998074751 3.04533077022745 3.3948320308852864 0.6824355007443828 0.586845
7029706842 3.8267144524467223 3.8566927851949706 2.0584258698239433 1.0645196863
05676 3.974022927550987 0.8577325222498544 0.9374132196884211 4.637251480865712
now, we have: 30 objects

E:\My Documents\IK407>

```

Figure 15 final testing5

9. Discussion

9.1 Project outcomes

9.1.1 What we did in this project?

In this project we combine K-clustering and multilevel technique, and in order to find this combination could be applied in data-base management and also in searching information in internet, which could make both of them more effective.

According to our result we achieve our initial purpose of cluster objects and choose the best data-base we want. In this course we use 1000 data as our object, after we cluster them and use fractional knapsack to find the best data in the entire objects. According to our evaluation we know that all these data we finally get is exactly what we want.

But in our programming course we were went to a wrong way when we were using fractional knapsack in choose the best data. We thought when we use other data to exchange what we put in our “knapsack”; this course might be random but actually it is not. It should be a cycle, and in this course the algorithm should run $K*N$, which K is the size of the knapsack, and N is the sum of availability of objects.

9.1.2 What we can do in the future?

Maybe this combination work still has a lot of place need to improve, but we still believe this work has its significance. As we see we can use this in data-base management and webpage searching, so we guess we may make an extension of this and also could combine this project with pagerank technology to make the webpage searching more effective and more niceties.

We know today lot of searching technology is in a dynamic environment, but our work is in a static environment. So we can speculate that neither of them is perfect. Therefore, as we thought we can combine them two together to deal big data-base or web source. And if we go this way, I think we can have a very bright future in the way of searching engine's development.

9.2 Evaluation of the overall results

According to our experiment result, we find when we combine multilevel with K-clustering the result is different compared with we just use K-clustering.

As shown in the testing, we can see the solutions are not the same. Why is that? When we cluster the data with multilevel strategy, we drop the objects which outside the knapsack. We just get an initial solution of each cluster and we might drop the objects which can be available in the original cluster. That's the reason why multilevel clustering is not the precious solution.

But it is very useful in real life. When we fast the data from World Wide Web, we can't cluster them all in one time. And Multilevel is a way to implement it effective. And we can maintain it on database management. Also it can be used in computer workspace.

10. Conclusion

In this paper we try to achieve a combination of multilevel technique and K-clustering, we hope this work can applied in web page evaluation and data-base management. And this could be thinking an extension of searching engine technology. In our work we use multilevel technique to divide a big data-base into smaller one, and this can make our clustering work easier. And when we evaluate a webpage we also use a fractional knapsack as our solution. And this way is different than people did before, because our work is working in a static environment and before us they usually considered in a dynamic environment. In the future work, we can consider about how it works on stochastic situation, and use it in the webpage evaluated.

Appendix

A1 References

- [1] slides 50 the Lecture of Crawling written by Morten Goodwin Olsen
- [2] Ole-Christoffer Granmo, B.John Oommen, Svein A. Myrer and Morten G. Olsen, "Determining Optimal Polling Frequency Using a Learning Automata-based Solution to the Fractional Knapsack Problem," CIS, 2006
- [3] http://en.wikipedia.org/wiki/Data_clustering#K-means_clustering
- [4] http://en.wikipedia.org/wiki/Multilevel_models
- [5] <http://mathworld.wolfram.com/Stochastic.html>
- [6] <http://www.mydrs.org/dv7/dispbbs.asp?boardid=4&id=11335>
- [7] <http://blog.csdn.net/ljczy/archive/2006/03/19/629418.aspx>

The following references are the articles we read when we program and looking for something connect to the topic we write but don't used in the report.

- [8] http://en.wikipedia.org/wiki/Knapsack_problem
- [9] <http://www-cse.uta.edu/~holder/courses/cse2320/lectures/applets/knapsack/knapsack.html>
- [10] http://en.wikipedia.org/wiki/List_of_knapsack_problems
- [11] Y.s. Li, J.W. Lu, "The start of JAVA program," China Youth Electronic Publishing company, 2005
- [12] F.X. Zhu, "The design of JAVA program," WuHan University Publish, 2004