LUDMILA NAUMOVA

NP=P? Algorithms for solving NP-problems by matrix method in Scilab program

Ludmila Naumova

NP=P? Algorithms for solving NP-problems by matrix method in Scilab program

«Издательские решения»

Naumova L.

NP=P? Algorithms for solving NP-problems by matrix method in Scilab program / L. Naumova — «Издательские решения»,

ISBN 978-5-44-937302-1

We know the problems of combinatorics, such as the problem of permutations, combinations, placement, represented by the corresponding formulas. But these formulas only give us the number of solutions, not the solutions themselves. There were no common standard algorithms for solving these types of problems. These types of problems with large numbers can be referred to NP problems. But with the help of Scilab program typical algorithms of such problems are revealed and solutions are given.

ISBN 978-5-44-937302-1

Содержание

NP=P? Algorithms for solving NP-problems by matrix method in Scilab program

Ludmila Naumova

ISBN 978-5-4493-7302-1
Created with Ridero smart publishing system

Introduction

From the course of school mathematics, we all know the problems of combinatorics, such as the problem of permutations, combinations, placement, represented by the corresponding formulas. But these formulas only give us the number of solutions, not the solutions themselves. There were no common standard algorithms for solving these types of problems. These types of problems with large numbers can be referred to NP problems. But with the help of Scilab program typical algorithms of such problems are revealed and solutions are given, and not only the number of solutions. The essence of the algorithms consists in operating with elements of natural series in rows and columns of the matrix, as well as in operating with rows and columns of the matrix using Scilab's commands. NP-problems, in principle, represent all the same combinatorics problems, but in large numbers. So in one NP-problem can be present immediately as permutations and combinations and placement, can be these operations (combinations, placement, permutations) sequentially repeated, but with other, obtained in the course of solving the problem, data, can be set additional or any other conditions or calculations. The bottom line is that knowing the typical algorithms of permutations, placement, combinations, these algorithms can be used as much as necessary in a single problem and thus solve NP problems. Let us emphasize once again that the algorithms below give the solutions themselves, not just answers about the number of solutions, although they also give. In large numbers, the solution of these problems requires a large resolution power of the computer, but the algorithms remain the same. This book provides examples with small numbers, but the point remains the same. Thus, the author wants to show that often the solution of the problem lies on the surface, but sometimes we can not see the solution at this angle. The author is confident that more and more NP-problems will move into the category of P – problems. This process is inevitable with the development of programs and the growth of computer power.

Chapter 1. The essence of the method; commands and typical algorithms in the program Scilab 6.0.1

– The essence of the method

Any set can be written as a matrix with elements of this set. The essence of the method used consists of operating on natural numbers (elements of sets), applying a matrix approach, that is, operating on elements of matrices, their columns and rows, as well as in the interactions between matrices (sets).

Common algorithms for combinatorics problems, such as permutation, combination, placement problems, which are given below, are applicable for NP – problems. These types of problems (on permutations, combinations, placements) with large numbers can be attributed to NP – problems. NP-problems, in essence, represent all the same problems of combinatorics, but in a complicated version, in one problem can be present immediately as permutations and combinations and placement, these operations (combinations, placement, permutations) can be repeated sequentially, but with other data obtained in the course of solving the problem, additional conditions or calculations can be set. But with the help of Scilab program typical algorithms of such problems are revealed and solutions are given, not only the number of solutions. The bottom line is that knowing the typical algorithms of permutations, placement, combinations, they can be used as many as standard algorithms in one problem and thus solve NP – problems.

– NP-problems and their models in small numbers, General algorithms

Here are examples of NP-problems:

Task №1.

Suppose that you are organizing a group of four hundred University students. The number of places is limited, and only one hundred students will get a place in the hostel. The situation is complicated by the fact that the Dean has provided you with a list of pairs of students who can't live together, and asked that no pair from this list did not get to the final version.

Task №2.

Is it true that among the numbers {-2, -3, 15, 14, 7, -10, …} are there any such that their sum is 0?

Or else, for example: 50, 2, 47, 5, 21, 4, 78, 1. Problem: is it possible to choose among these numbers such that their sum will give 100?

Task №3.

You want to find the shortest path that passes exactly one time through each of the six cities A, B.C. D. I. F. You are given a matrix of distances between all pairs of cities,

Tasks like the above 3 examples seem unsolvable (so far no one has been able to prove that some of them are actually as complex as it seems, i.e. that there is really no way to get an answer using a computer).

We make models of these problems in small numbers to find an algorithm and solve these problems:

Task-model №1

Let's say you're hosting a group of 5 University students. The number of places is limited, and only 3 students will get a place in the hostel. The situation is complicated by the fact that the Dean provided you with a list of students who can't live together, and asked that none of this list was not in the final version.

Problem-model №1—1

Let's say you're hosting a group of 9 University students. The number of places is limited to 4 – 2 rooms for 2 people, and only 4 students will get a place in the hostel.

Find these solutions.

Task-model №3.

You want to find the shortest path that passes exactly once through each of the four cities A, B. C. D.. You are given a matrix of distances between all pairs of cities,

Solutions of NP-problems and their model problems are similar and have the same algorithms, solutions of model problems are given below.

– Specify the source data

The initial data in the commands are given by a vector (a single matrix), but it is also possible by a two-dimensional matrix, several matrices, etc. each object is assigned a serial number. For example, we have 5 students:
Let's give each student a number in order: 1. – first student; 2. – 2nd student....etc to 5. As a one-dimensional matrix n
Program start:
loading the source environment
– > n= [1 2 3 4 5]
n =
1. 2. 3. 4. 5.

– Permutations

The permutation is performed using the perms (n) command:

Now we find all possible permutations from 1 to 5, there will be 120. The answer will be written in the form of a matrix, where each row is a variant of one of the permutations, the number of rows in the matrix will be equal to the number of permutations, and the number of columns will be equal to the originally specified elements (in our case 5).

– > P=perms (n)

– Permutations followed by matrix replacement and finding solutions

As an example with a complex permutation (replacement of the matrix obtained as a permutation to another matrix), the problem-model №3:

You want to find the shortest path that passes exactly once through each of the four cities A, B. C. D. You are given a matrix of distances between all pairs of cities,

Decision.

The essence of the solution is that finding all the permutations between the four cities in the form of rows of the matrix, replace the rows of the resulting matrix with the rows of another matrix, the elements of which are the distances between the cities and calculate the paths, then find the smallest.

Set the initial conditions: cities A, B,C, D are numbered in order and assign each city number 1,2,3,4, respectively. Let's set the distance between cities by matrices, for example. distance between town A and B as a matrix ab whose elements is a pair of 1 and 2 (this is the number of cities A and B):

– > ab= [1 2];
– > ac= [1 3];
– > ad= [1 4];
– > ba= [2 1];
– > bc= [2 3];
– > bd= [2 4];
– > ca= [3 1];
– > cb= [3 2];
– > cd= [3 4];
– > da= [4 1];
– > db= [4 2];
– > dc= [4 3];
– > M= [1 2 3 4]
M =
– 2. 3. 4.

We find all possible permutations and obtain the matrix P.

– -> P=perms (M);

The result is a matrix of 4 columns (cities) and rows-variants of permutations.

If in the condition of the problem it was necessary to return back to the initial point, then to the matrix obtained as a result of permutations it would be necessary to add another one column where the element in each row would be the element of the first row of the matrix P.

The program does not provide a command to replace the original matrix, therows of which are the paths indicated by a sequential enumeration of cities, with a matrix of distances between these cities (for example, such a command could be called command "between". The value of command "between" the elements with values 1 and 2 is 10, for example, as the initial data between ([1 2]) =10; insert values between the elements of the matrix rows P as between (P:,1)). So we'll have to go the other way. Divide the resulting matrix P into 3 parts, and then connect again, as between 4 cities can build a path of three distances between cities. These 3 matrices will consist: the 1st of the first two columns, the 2nd of the second and third columns, the 3rd of the third and fourth columns.

– > N=P;
– > N (:,4) = [];
– > N (:,3) = [];
– > A=N;

```
– > X=P;
– > X (:,1) = [];
– > X (:,3) = [];
```

Конец ознакомительного фрагмента.