Людмила Наумова

NP=P? Алгоритмы решения NP-задач матричным методом в программе Scilab

Людмила Наумова

NP=P? Алгоритмы решения NP-задач матричным методом в программе Scilab. Математическое эссе

«Издательские решения»

Наумова Л.

NP=P? Алгоритмы решения NP-задач матричным методом в программе Scilab. Математическое эссе / Л. Наумова — «Издательские решения»,

ISBN 978-5-44-937193-5

Из курса школьной математики нам все известны задачи комбинаторики, такие как задачи на перестановки, сочетания, размещения. NP- задачи, в принципе, представляют все те же задачи комбинаторики, но в больших числах.

ISBN 978-5-44-937193-5

Содержание

NP=P? Алгоритмы решения NP-задач матричным методом в программе Scilab Математическое эссе

Людмила Наумова

ISBN 978-5-4493-7193-5
Создано в интеллектуальной издательской системе Ridero

Введение

Из курса школьной математики нам все известны задачи комбинаторики, такие как задачи на перестановки, сочетания, размещения, представленные соответствующими формулами. Но эти формулы дают нам только количество решений, а не сами решения. Общих типовых алгоритмов самих решений на эти типы задач не было. Эти типы задач с большими числами можно отнести к NP задачам. Но с помощью программы Scilab раскрываются типовые алгоритмы таких задач и выдаются сами решения, и не только количество решений. Суть алгоритмов состоит в оперировании с элементами натурального ряда в строках и столбцах матрицы, а также в оперировании со строками и столбцами матрицы с помощью команд Scilab. NP- задачи, в принципе, представляют все те же задачи комбинаторики, но в больших числах. Так в одной NP- задаче могут присутствовать сразу как перестановки, так и сочетания и размещения, могут быть эти операции (сочетания, размещения, перестановки) последовательно повторяться, но уже с другими, полученными в ходе решения задачи, данными, могут быть заданы дополнительные еще какие либо условия или вычисления. Суть в том, что зная типовые алгоритмы перестановок, размещения, сочетания, эти алгоритмы можно использовать сколько угодно в одной задаче и таким образом решать NP задачи. Подчеркнем еще раз, что приведенные ниже алгоритмы дают сами решения, а не только ответы о количестве решений, хотя и их тоже дают. В больших числах решение этих задач требует большой разрешительной мощности компьютера, но алгоритмы остаются все те же. В данной книге приводятся примеры с малыми числами, но суть дела остается та же. Тем самым, автор хочет показать, что часто решение задачи лежит на поверхности, но мы иногда не можем увидеть решение под таким углом. Автор уверен, что все больше NP-задач будет переходить в разряд P- задач. Этот процесс неизбежен с развитием программ и ростом мощности компьютеров.

Глава 1. Сущность метода, команды и типовые алгоритмы в программе Scilab 6.0.1

– Сущность метода

Любое множество можно записать в виде матрицы с элементами этого множества. Сущность применяемого метода состоит из оперирования над натуральными числами (элементами множеств), применяя матричный подход, то есть оперированием элементами матриц, их столбцов и строками, а также во взаимодействиями между матрицами (множествами).

Общие типовые алгоритмы для задач комбинаторики, таких как задач на перестановки, сочетания, размещения, которые приведены ниже, применимы для NP- задач. Эти типы задач (на перестановки, сочетания, размещения) с большими числами можно отнести к NP- задачам. NP- задачи, по сути своей, представляют все те же задачи комбинаторики, но в усложненном варианте, в одной задаче могут присутствовать сразу как перестановки, так и сочетания и размещения, могут быть эти операции (сочетания, размещения, перестановки) последовательно повторяться, но уже с другими полученными в ходе решения задачи данными, могут быть заданы дополнительные еще какие либо условия или вычисления. Но с помощью программы Scilab раскрываются типовые алгоритмы таких задач и выдаются сами решения, а не только количество решений. Суть в том, что зная типовые алгоритмы перестановок, размещения, сочетания, их можно использовать сколько угодно как типовые алгоритмы в одной задаче и таким образом решать NP- задачи.

– NP-задачи и их модели в малых числах, общие алгоритмы

Приведем примеры NP-задач:

Задача №1.

Предположим, что вы организуете размещение группы из четырехсот студентов университета. Количество мест ограничено, и только сто студентов получат места в общежитии. Ситуация усложняется тем, что декан предоставил вам список пар студентов, которые не могут жить вместе, и просил, чтобы ни одна пара из этого списка не попала в окончательный вариант.

Задача №2.

Верно ли, что среди чисел {—2, —3, 15, 14, 7, —10, …} есть такие, что их сумма равна 0?

Или еще, например, примерно такая же задача: 50, 2, 47, 5, 21, 4, 78, 1. Задача: можно ли подобрать среди этих чисел такие, что их сумма даст 100?

Задача №3.

Требуется найти кратчайший путь, проходящий точно по одному разу через каждый из шести городов A, B. C. D.I. F_6. Задана матрица расстояний между любыми парами городов,

Задачи, подобные по приведенным выше 3-м примерам кажутся неразрешимыми (до сих пор никто не смог доказать, что какая-то из них на самом деле так сложна, как кажется, т.е. что действительно нет возможности получить ответ с помощью компьютера).

Составим модели этих задач в малых числах для нахождения алгоритма и решения этих задач:

Задача-модель №1

Предположим, что вы организуете размещение группы из 5 студентов университета. Количество мест ограничено, и только 3 студента получат места в общежитии. Ситуация усложняется тем, что декан предоставил вам список студентов, которые не могут жить вместе, и просил, чтобы никто из этого списка не попал в окончательный вариант.

Задача-модель №1—1

Предположим, что вы организуете размещение группы из 9 студентов университета. Количество мест ограничено 4 – это 2 комнаты по 2 человека, и только 4 студента получат места в общежитии.

Найти эти решения.

Задача-модель №3.

Требуется найти кратчайший путь, проходящий точно по одному разу через каждый из четырех городов A, B. C. $D._6$. Задана матрица расстояний между любыми парами городов,

Решения NP-задач и их задач-моделей аналогичны и имеют одни и те же алгоритмы, решения задач-моделей приведены ниже.

– Задание исходных данных

Исходные данные в командах задаются вектором (единичной матрицей), но возможно и двумерной матрицей, несколькими матрицам и т. д. Каждому объекту присваивается порядковый номер. Например, имеем 5 студентов:

Зададим каждому студенту свой номер по порядку: 1.-первый студент; 2.– 2-й студент….и т. д. до 5. В виде одномерной матрицы n

Запуск программы:

загрузка исходного окружения

– > n= [1 2 3 4 5]

n =

– 2. 3. 4. 5.

– Перестановки

Перестановка осуществляется при помощи команды perms (n):

Теперь найдем все возможные перестановки от 1 до 5-ти, их будет 120. Ответ запишется в виде матрицы, где каждая строка – это вариант одной из перестановок, число строк в матрице будет равно количеству вариантов перестановок, а число столбцов будет равно исходно заданным элементам (в нашем случае 5).

– > P=perms (n)

Перестановки с последующей заменой матрицы и нахождения решений

Как пример со сложной перестановкой (замена матрицы, полученной как перестановки на другую матрицу), задача-модель №3:

Требуется найти кратчайший путь, проходящий точно по одному разу через каждый из четырех городов A, B. C. D.[6]. Задана матрица расстояний между любыми парами городов,

Решение.

Сущность решения состоит в том, что найдя все перестановки между четырьмя городами в виде строк матрицы, заменяем строки полученной матрицы строками другой матрицы, элементами которой являются расстояния между городами и вычисляем пути, затем находим наименьший.

Зададим начальные условия: города A, B, C,D пронумеруем по порядку и присвоим каждому городу номер 1,2,3,4 соответственно. Зададим расстояние между городами матрицами, например. расстояние между городом A и B как матрицу ab, элементами которой является пара 1 и 2 (это номера городов A и 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.

Найдем все возможные варианты перестановок и получим матрицу P.

– -> P=perms (M);

Получилась матрица из 4-х столбцов (городов) и строк – вариантов перестановок.

Если бы в условии задачи надо было вернуться обратно в исходный пункт, то к полученной в результате перестановок матрице надо было бы добавить еще 5-йстолбец, где элементом в каждой строке которого, стоял бы первый элемент строки матрицы P.

В программе не предусмотрена команда замены исходной матрицы, строки которой — это пути, обозначенные последовательным перечислением городов, на матрицу расстояний между этими городами (К примеру, такую бы команду можно было бы назвать between. Значение между элементами со значениями 1 и 2 равно 10, к примеру, как исходные данные between ([1 2]) =10; вставка значений между элементами строк матрицы P как between (P:,1)). Поэтому придется пойти обходным путем. Разделим полученную матрицу P на 3 части, а затем снова соединим, так как между 4-мя городами можно построить путь из трех расстояний между городами. Эти матрицы будут состоять:1-я из первых двух столбцов, 2- я из второго и третьего столбца, 3-я – из третьего и четвертого столбца.

– > N=P;

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

Конец ознакомительного фрагмента.