Оптимальное упорядочение конфликтующих объектов и задача коммивояжера

А.В. Воеводин, С.А. Косяченко
Silver-AVV@yandex.ru , spiero@yandex.ru
Видео Интернешнл

Аннотация. В работе представлена постановка задачи оптимального упорядочения конфликтующих объектов и ее связь с задачей коммивояжера (Travelling Salesman Problem или TSP). Задача оптимального упорядочения конфликтующих объектов возникает в социологии, при анализе графов в социальных сетях, при размещении рекламных заказов в сетях СМИ. В статье описаны используемые авторами на практике быстрые алгоритмы решения этой и связанных с ней задач. Также рассмотрена задача TSP с разреженной матрицей штрафов. Для задач TSP с ленточной и блочно-диагональной матрицами найдены необходимые и достаточные условия и построены точные алгоритмы, при которых достигается нулевое минимальное значение целевой функции задачи. Предложены эффективные алгоритмы и для произвольных разреженных матриц. Приведены результаты аналитических и численных исследований сложности разработанных алгоритмов и точности решения, а также рекомендации по применению алгоритмов для решения задач подобного типа.

Ключевые слова: оптимальное размещение; задача коммивояжера; TSP; NP-трудные задачи; ленточные матрицы; разреженные матрицы; жадный алгоритм; штрафная функция; конфликты, сети СМИ.

Введение

Задача оптимального размещения конфликтующих объектов обобщает задачу коммивояжера, такие задачи возникают во многих предметных областях. Это классическая задача коммивояжера обхода всех городов по кратчайшему пути [1], задача о сверлильном станке, задача о производстве красок, пополнение банкоматов наличными деньгами, сбор сотрудников вахтовым методом, расклейка афиш, другие задачи логистики [2]. В общем случае в рассматриваемой нами задаче не все объекты связаны и конфликтуют между собой, а назначаемые за конфликт штрафы могут быть большими при непосредственном соседстве объектов и уменьшаются при отдалении их друг от друга в списке. Для задач коммивояжера обычно существует один тип

штрафа, например: расстояние, стоимость или время доставки груза между городами. В общем случае может быть несколько типов штрафов, определяемых разными причинами нежелательного соседства объектов, например, принадлежностью их к конфликтующим группам по различным признакам. Такие задачи возникают в социологии, при нахождении оптимальных путей между вершинами графа в социальных сетях, при размещении рекламных заказов в сетях СМИ. Несмотря на то, что данная задача в общем случае относится к классу NP-трудных задач, для наиболее распространенных частных случаев задачи TSP с разреженной матрицей штрафов построены точные быстрые алгоритмы. Для общего случая предложен эвристический быстрый алгоритм, масштабируемый до произвольного размера при сохранении линейной сложности при незначительных потерях в качестве решения. Это позволяет нам практически применять предложенные алгоритмы на больших объемах данных.

Постановка задачи и ее связь с задачей коммивояжера

Дано неупорядоченное множество из N объектов x_i, $i=1,..,N$. Требуется упорядочить эти объекты в виде линейного списка. Объекты «конфликтуют» между собой, конфликтующие объекты желательно разместить в списке подальше друг от друга. Математически конфликт выражается в том, что за непосредственное соседство в списке объектов x_i и x_j применяется штраф $p_{ij} \geq 0$. Если же между x_i и x_j в списке находится еще один объект x_k, то штраф за соседство x_i и x_j уменьшается в 2 раза. Если между x_i и x_j в списке находятся два других объекта, то штраф за соседство x_i и x_j уменьшается в 3 раза и т.д. В общем случае если после упорядочения в списке между объектами x_i и x_j оказалось $L_{ij} \geq 0$ других объектов, то за такую близость x_i и x_j применяется штраф, равный $\frac{p_{ij}}{L_{ij}+1}$. К примеру, при N=4 перестановке объектов x_3, x_1, x_4, x_2 будет соответствовать суммарный штраф

$$W = p_{31} + p_{14} + p_{42} + (p_{34} + p_{12})/2 + p_{32}/3.$$

Требуется среди всех перестановок объектов найти перестановку с минимальной величиной штрафа W. Формально постановка задачи выглядит следующим образом. Имеется множество из N объектов $x_1, x_2, \dots, x_N$. На исходной нумерации этих объектов задается матрица штрафов P с элементами $p_{ij} \geq 0$ за непосредственное соседство в списке объектов x_i и x_j. Пусть S – множество всех перестановок s_k чисел 1, 2,..., N, где $k=1,\dots, N!$. Обозначим $L_{ij}^k \geq 0$ - количество объектов из исходного множества, оказавшихся в перестановке s_k между объектами x_i и x_j. Необходимо среди всех перестановок s_k объектов найти перестановку с минимальным суммарным штрафом W_k:

$$\min_k W_k = \min_k \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{p_{ij}}{L_{ij}^k + 1} \qquad (1)$$

В частном случае, когда штрафуется только непосредственное соседство объектов, все значения $L_{ij}^k = 0$, и штраф p_{ij} применяется только *N* - 1 раз к непосредственно соседствующим объектам. В этом случае задачу (1) упорядочения конфликтующих объектов можно записать в виде

$$\min_k W_k = \min_k \sum_{i=1}^{N-1} p_{i,i+1} \qquad (2)$$

где минимум берется по всем *N!* перестановкам s_k индексов объектов x_i, *i = 1, 2,..., N*.

Оптимизационная задача (2) является классической задачей TSP, которая в теории сложности относится к классу NP-трудных задач. Для метрических задач TSP существуют приближенные алгоритмы полиномиальной сложности, которые гарантированно находят решение максимум вдвое отличающееся от оптимального[1]. Рассматриваемая нами задача (2) относится к варианту незамкнутой, симметричной, неметрической задачи TSP, для которых пока не предложено приближенных алгоритмов полиномиальной сложности с хорошей оценкой точности. Таким образом, в общем случае для задач (1) и (2) не приходится рассчитывать на существование полиномиально быстрых и точных алгоритмов решения. Однако для ряда распространенных на практике случаев нам удалось это сделать.

Заметим, что хотя задача (2) оптимального упорядочения конфликтующих объектов формально относится к задаче TSP, фактически она отличается от классической задачи коммивояжера нахождения оптимального пути обхода городов на плоскости и требует построения специальных алгоритмов. Даже метрическую задачу TSP не всегда можно изобразить на плоскости как задачу обхода городов. Например, рассмотрим задачу с N равноудаленными друг от друга вершинами, т.е. все $p_{ij} = C > 0, i \neq j$, где C – константа. Такое множество вершин можно изобразить на плоскости только при *N*=3. В общем случае *N* равноудаленных вершин потребуют не менее (*N*-1)-мерного пространства, где образуют (*N*-1)-мерный регулярный симплекс [3]. Неметрическая задача TSP, к которой относится рассматриваемая нами задача (2), еще менее похожа на классическую задачу коммивояжера. В ней не выполняется неравенство треугольника, ее можно представить только в виде взвешенного графа или в матричном виде.

Но самое большое отличие задачи оптимального упорядочения конфликтующих объектов (2) от классической задачи коммивояжера заключается в том, что не каждый объект конфликтует с каждым, т.е. в

матрице штрафов некоторые значения $p_{ij} = 0$. Более того, взвешенный граф задачи (2) может быть несвязанным и может распадаться на несвязанные друг с другом компоненты. Поскольку наша задача неметрическая, то стоимость «обходного» пути между вершинами i и j может быть малой или даже нулевой: $p_{i,k} + p_{kj} = 0$, а стоимость «непосредственного» пути по дуге p_{ij} большой. Когда в матрице штрафов много нулевых элементов p_{ij}, она называется разреженной. Но несмотря на все эти отличия от классической задачи коммивояжера задача упорядочения конфликтующих объектов (2) остается задачей TSP и даже может иметь смысловое наполнение близкое к классическому. Матрица штрафов может означать стоимость дополнительных, не входящих в базовый набор, услуг на пути обхода городов, например: стоимость проезда по платным дорогам, оплата номеров и парковки в отелях повышенного класса на трассе и т.п. Такая матрица может быть разреженной. В данной статье мы построим алгоритмы, эффективные для задач TSP с разреженной матрицей штрафов. Будут предложены процедуры сведения задачи с хаотически разреженной матрицей к задачам с ленточной или блочно-диагональной матрицей. В свою очередь для задач TSP с ленточной или блочно-диагональной матрицами будут найдены необходимые и достаточные условия и построены алгоритмы, при которых всегда существует путь с $W^* = 0$. Важные и часто встречающиеся на практике классы задач с ленточной и блочно-диагональной матрицами рассмотрены в работе отдельно.

Матричная постановка задачи

В матричном виде задача (2) сводится к одноименной перестановке строк и столбцов симметричной неотрицательной матрицы стоимости $P = (p_{ij})$, $p_{ij} \geq 0$, так, чтобы сумма чисел на 2-й диагонали, находящейся непосредственно над главной, была минимальна. Любая одноименная перестановка строк и столбцов матрицы эквивалентна преобразованию $T*P*T^t$, где *T* – матрица перестановок, T^t – транспонированная к ней матрица. Матрица перестановок – это 0-1 матрица, у которой в любой строке и любом столбце только один ненулевой элемент. Перемножение матриц перестановок снова дает матрицу перестановок. Матрица T^t также является матрицей перестановок. Умножение слева матрицы *P* на матрицу *T* приводит к перестановке строк матрицы *P*. Умножение справа матрицы *P* на матрицу T^t приводит к одноименной перестановке столбцов матрицы *P*. Матрица $T*P*T^t$ снова будет симметричной матрицей. В результате одноименных перестановок строк и столбцов матрицы *P* ее диагональные элементы будут оставаться на главной диагонали. Таким образом, матричная постановка задачи (2) следующая. Найти матрицу перестановок *T* такую, чтобы у матрицы $T*P*T^t$ сумма чисел на 2-й диагонали была минимальна.

Оптимальный алгоритм для задачи с блочно-диагональной матрицей

Определение. Матрица, которая перестановкой одноименных строк и столбцов может быть приведена к блочно-треугольному виду, называется разложимой.

В нашем случае симметричной матрицы разложимая матрица приводится очевидно к блочно-диагональному виду. Симметричная матрица *P* разложима тогда и только тогда, когда ее граф несвязен. Справедлива следующая

Лемма 1. Если матрица *P* порядка *N* разложима и максимальный размер блока в ее блочно-диагольном представлении $\leq \left\lceil \frac{N}{2} \right\rceil$, то минимум в задаче (2) оптимального упорядочения объектов равен 0.

Приведем алгоритм, доказывающий лемму 1 и обеспечивающий оптимальное решение задачи (2).

Алгоритм 1. Оптимальное упорядочение строится следующим образом. Сначала в списке поочередно располагаем все первые вершины из каждого блока матрицы, затем вторые вершины из каждого блока матрицы и т.д. Поскольку при таком алгоритме любые две рядом стоящие в списке вершины *i* и *j* будут принадлежать разным диагональным блокам матрицы, они не будут связаны между собой, т.е. для любых соседних вершин $p_{ij} = 0$, а значит значение W^* суммы ЦФ в задаче (2) равно 0. Алгоритм требует *N*-1 шагов.

Следствие. Если граф задачи (2) несвязен, и максимальное число вершин в связной компоненте графа $\leq \left\lceil \frac{N}{2} \right\rceil$, то минимум в задаче (2) оптимального упорядочения объектов равен 0.

Отметим, что если в условиях леммы 1 использовать попарные перестановки вершин, то потребуется $\leq \left\lceil \frac{N}{2} \right\rceil$ попарных перестановок. Перестановке вершин *i* и *j* соответствуют попарные одноименные перестановки *i*-й и *j*-й строк и *i*-го и *j*-го столбцов матрицы *P*. В любой попарной одноименной перестановке строк и столбцов среди интересующих нас элементов 2-й диагонали матрицы *P* меняются только 4 элемента на новые. Точнее при перестановке вершин *i* и *j* элементы меняются следующим образом:

$$p_{i-1,i} \leftrightarrow p_{i-1,j}\,,\ p_{i,i+1} \leftrightarrow p_{i+1,j}\,,\ p_{j-1,j} \leftrightarrow p_{i,j-1}\,,\ p_{j,j+1} \leftrightarrow p_{i,j+1}\,. \tag{3}$$

Здесь мы учитываем, что матрица *P* симметричная. Таким образом, при попарной перестановк[illegible] достаточно в формуле суммы целевого функционала (2) подменить только [illegible]слагаемых. Если какой-либо из указанных индексов выходит из диапазона [illegible]N, то соответствующий элемент не переставляется и в сумме не участвует.

Алгоритм 1, обеспечивающий нулевой минимум в задаче (2) для разложимой матрицы и ее несвязанного графа, можно успешно применять в качестве приближенного алгоритма к матрицам близким к блочно-диагональным, у

которых все элементы, находящиеся вне диагональных блоков, малы относительно элементов внутри диагональных блоков.

Оптимальные алгоритмы для задачи с ленточной матрицей

Другим важным классом матриц, для которых можно построить алгоритм с нулевым значением штрафа в задаче (2), является класс ленточных симметричных матриц.

Определение. Матрица *P* порядка $N \geq 2$ называется ленточной, если

$$p_{ij} = 0, \qquad при\ \ |i-j| \geq l,$$

Величина $l > 0$ называется полушириной ленты такой матрицы.

Например, трехдиагональная матрица будет иметь полуширину ленты $l = 2$. Напомним, что в рассматриваемых нами задачах (1) и (2) матрица *P* симметричная.

Лемма 2. Если матрица *P* порядка *N* является ленточной с полушириной ленты $l \leq \left\lceil \frac{N}{2} \right\rceil - 1$, то минимум в задаче (2) оптимального упорядочения объектов равен 0.

Приведем алгоритм попарных перестановок строк и столбцов, обеспечивающий утверждение леммы 2.

Алгоритм 2. Для *N*≤5 утверждение очевидно, поскольку тогда $l \leq 1$ и матрица *P* имеет нулевую 2-ю диагональ. Пусть *N*>5. Индексы *i*, *j* попарных одноименных перестановок строк и столбцов будем брать как индексы элементов матрицы *P*, лежащие на ее $\left(\left\lceil \frac{N}{2} \right\rceil + 1\right) -$ й диагонали. Брать следует индексы не всех элементов с этой диагонали, а через один, начиная с 1-го. Число таких элементов будет $\left\lceil \frac{N}{4} \right\rceil$. Попарные перестановки выполняем последовательно в соответствии с индексами указанных элементов, начиная с 1-го верхнего элемента данной диагонали. При перестановках вершин *i* и *j* используем формулы (3). Из них видно, что мы последовательно переставляем четверки соседних с p_{ij} нулевых элементов, находящихся вне ленты матрицы, на ее 2-ю диагональ.

Отметим, что при попарных одноименных перестановках строк и столбцов с индексами *i*, *j* ленточной матрицы происходит заполнение элементов матрицы в *i*-й строке и *j*-м столбце, отстоящих от p_{ij} на $\leq l - 1$ позиций. Но этот факт не мешает работе данного алгоритма, поскольку индексы с $\left(\left\lceil \frac{N}{2} \right\rceil + 1\right) -$ й диагонали матрицы *P* берутся через один. Алгоритм 2 требует $\left\lceil \frac{N}{4} \right\rceil$ попарных перестановок одноименных строк и столбцов матрицы.

Ограничение на полуширину ленты в лемме 2 можно ослабить, если использовать алгоритм с бóльшим числом перестановок. А именно, справедливо следующее утверждение.

Лемма 3. Если матрица P порядка N является ленточной с полушириной ленты $l \le \left[\frac{N}{2}\right]$, то минимум в задаче (2) оптимального упорядочения объектов равен 0.

Приведем алгоритм, обеспечивающий утверждение леммы 3.

Алгоритм 3. Оптимальное упорядочение строится следующим образом. Рассмотрим упорядочение вершин 1,2,… N, задаваемое исходной ленточной матрицей. В оптимальном списке располагаем эти вершины подряд в каждую четную позицию, начиная со 2-й позиции. Достигнув конца списка, вершины располагаем подряд в каждую нечетную позицию, начиная с 1-й позиции. Оптимальное упорядочение построено. Например, для $N = 8$ максимальная полуширина ленты матрицы P будет $l = 4$. Оптимальное упорядочение вершин будет таким: 5,1,6,2,7,3,8,4. Такое упорядочение обеспечивает нулевой минимум в задаче (2), ибо из структуры исходной ленточной матрицы следует, что в построенном оптимальном списке любые две рядом стоящие вершины i и j не связаны между собой, так как для них $p_{ij} = 0$. Так, в рассматриваемом примере в исходной ленточной матрице элементы $p_{51} = p_{16} = p_{62} = p_{27} = p_{73} = p_{38} = p_{84} = 0$. Значит, для указанного оптимального списка вершин 5,1,6,2,7,3,8,4 минимум в задаче (2) равен 0. Алгоритм 3 требует N-1 шагов – примерно в 4 раза больше, чем алгоритм 2.

Следствие. Если для графа задачи (2) существует упорядочение вершин, при котором любая вершина в списке связана только с вершинами, отстоящими от нее не более, чем на $\left[\frac{N}{2}\right] - 1$ позиций в списке, то минимум в задаче (2) оптимального упорядочения объектов равен 0.

Действительно, такое упорядочение вершин описывается ленточной матрицей с полушириной ленты $\le \left[\frac{N}{2}\right]$. Значит, в силу леммы 3, $W^* = 0$.

Отметим, что указанное в лемме 3 ограничение на полуширину ленты $\le \left[\frac{N}{2}\right]$ является предельным. Если полуширина ленты превышает это значение, то нулевое значение функционала задачи (2) гарантировать уже нельзя. Точнее справедлива следующая

Теорема 1. Пусть P - симметричная ленточная матрица порядка N, у которой все $p_{ij} > 0$ ($i \ne j$) в пределах ее ленты. Для того чтобы минимум в задаче (2) оптимального упорядочения объектов был равен 0, необходимо и достаточно, чтобы полуширина ее ленты $l \le \left[\frac{N}{2}\right]$.

Доказательство. Достаточность условия следует из леммы 3. Докажем необходимость от противного: если $l > \left[\frac{N}{2}\right]$, то $W^* > 0$. Пусть полуширина ленты $l \ge \left[\frac{N}{2}\right] + 1$. Тогда, поскольку ширина ленты L=2*l-1, получаем, что $L \ge 2 * \left[\frac{N}{2}\right] + 1 \ge N$. То есть ширина ленты $L = N$. А по условиям теоремы все $p_{ij} > 0$ ($i \ne j$) в пределах ленты. Значит в матрице P существует строка с некоторым номером k, в которой все ее элементы $p_{kj} > 0$ для $j \ne k$ от 1 до N, и столбец с некоторым номером k, в котором все его элементы $p_{ik} > 0$ для $i \ne k$ от 1 до N. Другими словами вершина с номером k конфликтует со всеми другими вершинами. А поскольку при любых перестановках вершина с номером k будет соседствовать хотя бы с одной другой вершиной, то для фунционала в задаче (2) будет справедлива оценка

$$W^* \ge \min_{i \ne k} p_{ik} > 0,$$

что и требовалось доказать.

Алгоритмы 2 и 3, обеспечивающие нулевой минимум в задаче (2), можно успешно применять в качестве приближенных алгоритмов к матрицам близким к ленточным, у которой все элементы, находящиеся вне ее ленты, малы относительно элементов внутри ленты. Для приведения матрицы к ленточной или близкой к ленточной можно применять специальные алгоритмы.

Алгоритмы для задач с разреженными и матрицами общего вида

Для случая разреженных матриц общего вида, когда в матрице P ненулевые элементы расположены хаотично, можно использовать эффективные методы группирования ненулевых элементов на диагоналях близких к главной. Например, это прямой и обратный методы Катхилл-Макки для разреженных матриц, которые с помощью одноименных перестановок строк и столбцов приводят матрицу к ленточному виду с шириной ленты значительно меньшей, чем у исходной. Сложность этой процедуры $O(N^2)$ [4]. В работах [4,5] приводятся многочисленные примеры, когда, применяя прямой или обратный методы Катхилл-Макки к разреженной матрице размера N, получаем ленточную матрицу с шириной ленты в несколько раз или даже на порядок меньше, чем у исходной матрицы. А согласно теореме 1 для достижения нулевого минимума функционала достаточно всего лишь, чтобы полуширина ленты была $\le \left[\frac{N}{2}\right]$. В этом случае, после процедуры Катхилл-Макки следует применить алгоритмы 2 или 3 оптимального упорядочения для достижения нулевого минимума функционала задачи (2).

Этот же подход можно успешно применять и в случаях, когда матрица *P* не является разреженной, то есть *P* – матрица общего вида, но в ней явно выделяются большие и малые элементы. Для матриц такого вида также сначала используем методы Катхилл-Макки для группирования больших элементов на диагоналях близких к главной. Затем применяем алгоритмы 2 или 3 оптимального упорядочения для минимизации функционала задачи (2). Таким образом мы построили новый быстрый алгоритм приближенного решения классической общей задачи TSP. Этот алгоритм будет достаточно точно работать в тех случаях, когда элементы p_{ij} в матрице штрафов *P* достаточно сильно отличаются по величине друг от друга, а число «малых» элементов в матрице не меньше $N^2/4$.

Алгоритмы для задачи с матрицей общего вида

Несмотря на существование оптимальных быстрых алгоритмов для блочно-диагональных, ленточных и разреженных матриц, а также эффективных алгоритмов для близким к ним матриц, в общем случае в задачах (1) и (2) не приходится рассчитывать на достаточно точные алгоритмы полиномиальной сложности. Сложность алгоритма полного перебора составит *O(N!)*. Если для малых *N* выполнение полного перебора возможно, то с ростом *N* мы столкнёмся с проблемой нехватки вычислительных ресурсов. Ограничения в вычислительных ресурсах приводят к необходимости построения экономичного в вычислительном смысле алгоритма поиска решения, близкого к оптимальному.

Применим к решению задач “жадный” алгоритм [6] (greedy algorithm), в котором каждый шаг является локально оптимальным. Рассмотрим работу жадного алгоритма для поставленной выше задачи (1) размещения на примере.

Алгоритм 4. Пусть есть множество объектов (x_1, x_2, x_3, x_4). Матрица попарных штрафов имеет вид: $P = \begin{vmatrix} 0 & 1 & 2 & 2 \\ 1 & 0 & 3 & 3 \\ 2 & 3 & 0 & 1 \\ 2 & 3 & 1 & 0 \end{vmatrix}.$

Так как в работе жадного алгоритма нам необходимо начинать размещение с самых конфликтных объектов, упорядочим объекты в порядке убывания «конфликтности», которую определим как сумму элементов соответствующей строки матрицы *P*. Получим вектор конфликтности $\bar{p}$=(5,7,6,6). В соответствии с ним будем размещать объекты в порядке (x_2, x_3, x_4, x_1).

Ищем оптимальное размещение для x_2, ставим x_2 в первую доступную позицию:

$x_2$2			

С учетом уже поставленного x_2 ищем оптимальное размещение для x_3:

$x_2$2			$x_3$3

С учетом уже поставленных x_2 и x_3 ищем оптимальное размещение для x_4:

$x_2$2	$x_4$4		$x_3$3

Наконец, ставим x_1 в последнюю доступную позицию:

$x_2$2	$x_4$4	$x_1$1	$x_3$3

Получаем штраф *W*=9.

Основным недостатком «жадного» алгоритма является оптимизация только одного следующего шага. Для его улучшения авторы предлагают воспользоваться следующим эвристическим алгоритмом.

Алгоритм 5.

1. Если *N*≤*7*, то выполняем оптимальное размещение полным перебором и на выход. Иначе переходим к шагу 2.
2. Все объекты сортируются в порядке убывания «конфликтности». Пронумеруем их заново $(x_1, x_2, \ldots, x_N)$. Смещение *S* = 0. Список размещения – пустой.
3. Берем 7 объектов $(x_{S+1}, \ldots, x_{S+7})$ и с помощью полного перебора ищем оптимальное размещение этого подмножества в списке.
4. Если *S+7=N*, то все объекты размещены - Выход.
5. Из всех размещенных 7 объектов оставляем в списке только первый x_{S+1}, остальные убираем.
6. *S = S + 1*. Возвращаемся к шагу 2.

Отличием алгоритма 5 от приведенного выше жадного алгоритма 4 заключается в том, что оптимальное размещение x_{S+1} ищется не локально оптимально, а с учетом других конфликтующих объектов, что позволяет лучше определять оптимальную позицию для x_{S+1}.

Если алгоритм 5 применить к вышеприведенному примеру, то выполняя полный перебор даже для 2 объектов вместо 7, получим оптимальное значение суммарного штрафа *W*=7.5.

В данном случае нам удалось добиться глобального минимума, хотя понятно, что в общем случае данный алгоритм на такой результат не претендует.

Как показывает практика, предложенный алгоритм 5 удовлетворительно решает задачи размером *N* ≤ 30. При *N*>30 скорость работы алгоритма начинает заметно падать. Для компенсации этого недостатка для больших *N* предлагается разбивать позиции списка на подблоки таким образом, чтобы длина каждого подблока была не более 30, и проводить оптимизацию в каждом подблоке «почти» отдельно.

Алгоритм 6. Имеем N>30 объектов, они отсортированы в порядке убывания штрафа $(x_1, x_2, \ldots, x_N)$. Вычисляем количество подблоков $K=\left\lceil \frac{N}{30} \right\rceil$. Начиная с первого объекта, распределяем последовательно все объекты $(x_1, \ldots, x_N)$ поочередно по подблокам 1,...,K. Когда подблоки кончатся, распределяем объекты с последнего подблока до первого, затем снова с первого до последнего и т.д. Таким образом, в каждом подблоке мы получим примерно одинаковый по конфликтности набор объектов. После такого разбиения предложенный выше алгоритм размещения применяем к каждому подблоку отдельно. А для того, чтобы избежать размещения конфликтующих объектов на границе подблоков будем применять приведенный выше алгоритм еще и на границе подблоков следующим образом.

Рассмотрим пример. Пусть имеется 2 подблока. Все множество объектов соответственно поделено на 2 группы объектов, которые должны быть размещены в подблоках 1 и 2 соответственно. Сначала выполняется размещение в 1-м подблоке. Затем размещаем объекты 2-ого подблока с учетом размещения 1-ого подблока. Затем выделяем промежуточный подблок на границе 1-ого и 2-ого подблока. В данный промежуточный подблок входят $\left\lceil \frac{l1}{2} \right\rceil$ последних объектов 1-ого подблока и $\left\lceil \frac{l2}{2} \right\rceil$ первых объектов 2-ого подблока, где $l1$, $l2$ – количество объектов в 1-ом и во 2-ом подблоке соответственно. Выполняем размещение в промежуточном подблоке с учетом расстановки первых $l1 - \left\lceil \frac{l1}{2} \right\rceil$ объектов 1-ого подблока. Далее еще раз проводим переразмещение объектов во 2-ом подблоке. Таким образом, удается улучшить размещение на границе подблоков. Заметим, что результат работы предложенного алгоритма 6 может быть далек от оптимального, поэтому для конечной «шлифовки» будем дополнительно использовать алгоритм локальной оптимизации, приведенный ниже.

Алгоритм локальной оптимизации с помощью попарных перестановок

Будем менять местами все пары вершин (x_i, x_j), если это приводит к уменьшению ЦФ. Так поступаем до тех пор, пока происходит уменьшение ЦФ. Можно также на каждом шаге выбирать наилучшую перестановку вершин i и j, у которой (сумма старых 4 элементов 2-й диагонали) - (сумма новых 4 элементов 2-й диагонали) максимальна. При этом учитываем, что в любой попарной одноименной перестановке строк и столбцов среди интересующих нас элементов 2-й диагонали матрицы P меняются только 4 старых элемента на новые. Индексы старых и новых элементов определяются формулами (3). Так поступаем, пока наилучшая перестановка будет давать

выигрыш по функционалу. Такой метод сходится к локальному минимуму за конечное число шагов. Сложность одного шага $O(N^2)$. Если такую процедуру попарной перестановки применять только внутри каждого подблока, то общая сложность всех попарных перестановок и предложенного алгоритма упорядочения всех конфликтующих объектов в целом будет равна $O(N)$.

Общая схема функционирования предложенного алгоритма приведена на рисунке 1.

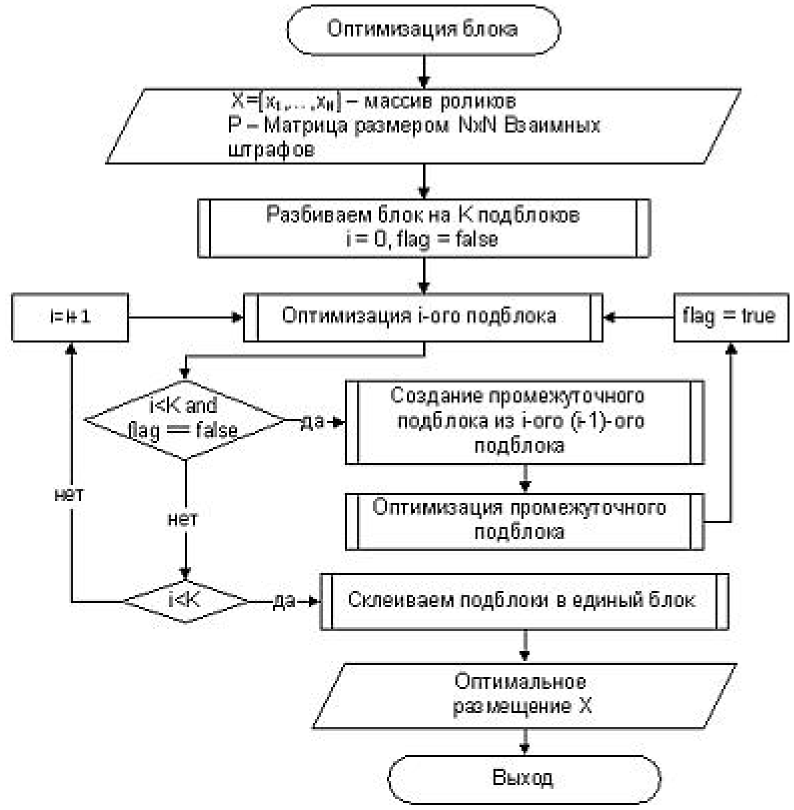


Рис. 1.

Сложность предложенного алгоритма в целом линейна и равна $C*N/30$, где C – число операций, приходящихся на обработку одного подблока размера не более 30.

Численные исследования

Оценим численно точность и скорость работы предложенного выше алгоритма. Определить точность работы можно, зная оптимальное значение ЦФ, которое можно получить алгоритмом полного перебора. Ввиду большой ресурсоемкости такого подхода сделать это представляется возможным только для небольших N. Результаты сравнительных исследований для $N \leq 10$ приведены в таблице 1. В колонке «Объекты» в таблице 1 приведена кодировка набора объектов, на котором проводился замер точности работы алгоритмов. Например, кодировка «1112223» обозначает, что имеется 7 объектов, которые поделены на 3 подгруппы: три одинаковых объекта «1», три одинаковых объекта «2» и объект «3». Каждый объект конфликтует только с объектами своей подгруппы с одинаковым единичным штрафом. Так как решение задачи (2) нахождения минимума W^* тривиально, в данной задаче мы искали только минимум W в задаче (1).

Таблица 1. Значения ЦФ для размещения 10 и менее объектов.

	Объекты	Алгоритм полного перебора. Значение ЦФ $W_{опт}$	Предложенный алгоритм. Значение ЦФ $W_{алг}$
7	1112223	2.58	2.58
8	11122334	2.26	2.26
9	111122233	3.82	4.02
10	1111222334	3.35	3.52

При больших значениях N нет возможности найти оптимальную расстановку с помощью алгоритма полного перебора, но можно провести специальные тесты с очевидным решением.

Тест 1. Возьмем множество из 100 объектов, которое состоит из двух подмножеств: $(x_1, x_3, x_5 \ \ldots x_{99})$, $(x_2, x_4, x_6 \ \ldots x_{100})$, по 50 одинаковых объектов в каждом. Каждый объект конфликтует только с объектами своего подмножества с одинаковым единичным штрафом. Будем искать минимум W в задаче (1). Оптимальная расстановка для данного теста совпадает с порядком следования объектов $x_1, \ldots x_{100}$ или, другими словами, оптимальным размещением будет являться чередование объектов из первого подмножества с объектами из второго подмножества. Подсчитаем значение $W_{опт}$ для задачи (1):

$$W_{опт}(x_1, x_2, x_3 \ \cdots x_{100}) = 2 \cdot W_{опт}(x_1, x_3, x_5 \ \cdots x_{99}) = 2 \cdot \begin{pmatrix} \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \cdots + \frac{1}{98} + \\ \frac{1}{2} + \cdots + \frac{1}{96} + \\ + \cdots + \\ + \frac{1}{2} \end{pmatrix} \qquad (4)$$

Здесь в первой строке записан штраф за конфликты объекта x_1 и всех остальных объектов первого подмножества $(x_3, x_5 \ \ldots x_{99})$. Во второй строке штраф за конфликты x_3 и всех остальных объектов первого подмножества, исключая x_1, и т.д. После преобразований формул (4), получим оптимальное значение функции штрафа

$$W_{опт} = 2 \cdot \sum_{i=1}^{49} \frac{(50 - i)}{2 \cdot i} = 174.96$$

Применяя к тому же входному множеству объектов предложенный алгоритм, получим $W_{алг}$ =175.86.

Тест 2. Возьмем набор из 100 объектов $x_1, x_2, \ldots x_{100}$. Матрицу попарных штрафов определим следующим образом:

$$\begin{cases} p_{ij} = 1, & \dfrac{|i-j|}{5} \in Z \\ p_{ij} = 0, & \dfrac{|i-j|}{5} \notin Z \end{cases}$$

Другими словами, из первоначального множества X конфликтуют между собой только элементы из 5-ти подмножеств: $(x_1, x_6, x_{11} \ \ldots x_{96})$, $(x_2, \ldots x_{97})$, $(x_3, \ldots x_{98})$, $(x_4, \ldots x_{99})$, $(x_5, \ldots x_{100})$. Оптимальная расстановка для данного теста совпадает с порядком следования объектов $x_1, \ldots x_{100}$.

Посчитаем оптимальный штраф для данного размещения. Ввиду того, что конфликтующие подмножества одинаковы с точки зрения штрафа, мы можем подсчитать штраф для одной группы, например $(x_1, \ldots x_{96})$, и умножить его на 5. Воспользовавшись определением ЦФ (1), получаем:

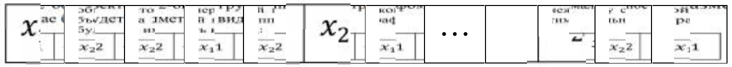
$$\text{И}_{\text{опт}}(x1, x2, \cdots x100) = 5 \cdot \text{И}_{\text{опт}}(x_1, x6 \cdots x96) = 5 \cdot \begin{pmatrix} \frac{1}{5} + \frac{1}{10} + \frac{1}{15} + \cdots + \frac{1}{95} + \\ \frac{1}{5} + \cdots + \frac{1}{90} + \\ + \cdots + \\ + \frac{1}{5} \end{pmatrix} \quad (4)$$

Здесь в первой строке записан штраф за конфликты x_1 и всех остальных объектов первого подмножества $(x_1, x_6, x_{11} \ \dots x_{96})$. Во второй строке штраф за конфликты x_6 и всех остальных объектов первого подмножества, исключая x_1, и т.д. После преобразований (4) получаем:

$$W_{\text{опт}} = 5 \cdot \sum_{i=1}^{19} \frac{(20-i)}{5 \cdot i} = 51{,}95$$

Применим к первоначальному подмножеству предложенный алгоритм, получаем $W_{\text{алг}}$ =52,76.

Тест 3. Возьмем 34 объекта, которые поделены на 2 неконфликтующие группы: 12 объектов первой группы конфликтуют между собой с штрафом *p*=10 и 22 объекта 2-ой группы с штрафом *p*=2. Оптимальное размещение в этом случае будет иметь вид:



где x_1 - объект из 1-ой группы, x_2 - объект из 2-ой группы.

Оптимальное значение ЦФ для такого размещения: $W_{\text{опт}}$=186,12.

Предложенный алгоритм находит решение: $W_{\text{алг}}$ =188,0.

Тест 4. Проведем тест аналогичный тесту 2, только с 3-мя группами по 30 взаимно не конфликтующих объектов. Т.е. имеем объекты $x_1, x_2, \dots x_{90}$. Функция попарных штрафов:

$$\begin{cases} p_{ij} = 1, & \frac{|i-j|}{3} \in Z \\ p_{ij} = 0, & \frac{|i-j|}{3} \notin Z \end{cases}$$

Оптимальная расстановка для данного теста совпадает с порядком следования объектов $x_1, \dots x_{90}$.

$$\text{И}_{\text{опт}} = 3 \cdot \text{И}_{\text{опт}}(x_1, x_4 \dots x_{87}) = 3 \cdot \begin{pmatrix} \frac{1}{3} + \frac{1}{6} + \frac{1}{9} + \cdots + \frac{1}{87} + \\ \frac{1}{3} + \cdots + \frac{1}{84} + \\ + \cdots + \\ + \frac{1}{3} \end{pmatrix} (4)$$

После преобразований получаем:

$$W_{\text{опт}} = 3 \cdot \sum_{i=1}^{29} \frac{(30-i)}{3 \cdot i} = 89{,}85$$

Применим к первоначальному подмножеству предложенный алгоритм, получаем $W_{\text{алг}}$ =90,38.

При подготовке данной статьи авторы пытались найти худшие примеры, в которых значение ЦФ предложенного алгоритма значительно отличалось бы от оптимального значения ЦФ, но это оказалось не так просто. Результаты численных исследований решения оптимизационной задачи (1) для ЦФ W и задачи (2) для ЦФ W^* приведены в таблице .

Табл. 2. Сводная таблица результатов численных исследований.

№ примера	$\overline{W_{\text{опт}}}$	$\overline{W_{\text{алг}}}$	$\frac{\overline{W_{\text{алг}}}-\overline{W_{\text{опт}}}}{\overline{W_{\text{алг}}}}*100$	$\overline{W^*_{\text{опт}}}$	$\overline{W^*_{\text{алг}}}$
1	174.96	175.86	0,51	0,0	3,0
2	51,95	52,76	1,54	0,0	0,0
3	186,12	188,0	1,00	18,0	18,00
4	89,85	90,38	0,59	0,0	0,0

Скорость работы предлагаемого алгоритма в секундах для больших *N* приведена на графике, изображенном на рис. 2.

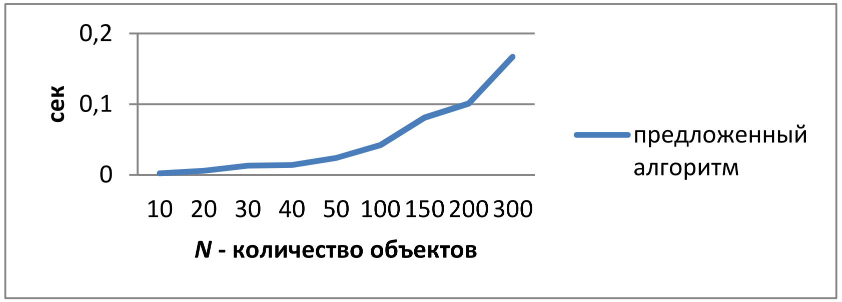


Рис. 1. Зависимость времени работы предложенного алгоритма от количества объектов

Сложность предложенного алгоритма упорядочения всех конфликтующих объектов в целом линейна и равна *O(N)*.

Заключение

В статье построены алгоритмы решения задач оптимального упорядочения конфликтующих объектов. Рассмотрены задачи двух типов: когда штрафуется только непосредственное соседство объектов (задача TSP), и обобщенный случай, когда штраф действует как на соседние, так и на отдаленные друг от друга конфликтующие объекты, и убывает в зависимости от числа расположенных между ними объектов. Показана связь задачи оптимального упорядочения конфликтующих объектов с задачей коммивояжера. Рассмотрена задача TSP с разреженной матрицей штрафов. Для наиболее распространенных случаев такой задачи TSP, когда матрица штрафов имеет вид ленточной или блочно-диагональной, построены точные быстрые алгоритмы, достигающие минимального нулевого значения целевой функции. Доказано необходимое и достаточное условие, при котором достигается нулевой минимум ЦФ в таких задачах TSP. Предложенные алгоритмы эффективны для разреженных матриц. Их также можно успешно применять в качестве приближенных алгоритмов к матрицам близким к ленточным, блочно-диагональным и разреженным. Для общего случая в статье предложен эвристический алгоритм, который показал высокое быстродействие при незначительных потерях в качестве решения. Для задачи TSP (2) он также работает эффективно. Удалось добиться хорошей его масштабируемости до произвольного размера при сохранении линейной сложности, что позволило нам использовать предложенные алгоритмы на практике на больших объемах данных, например, для задач размещения рекламных заказов в сетях СМИ. Их можно применять также для решения задач коммивояжера и близких задач обхода графов, в том числе в социологии, при нахождении оптимальных путей в социальных сетях.

Список литературы

[1]. Кузюрин Н.Н., Фомин С.А. Эффективные алгоритмы и сложность вычислений: Учебное пособие. – М.: МФТИ, 2007. - 312 с.

[2]. http://en.wikipedia.org/wiki/Travelling_salesman_problem

[3]. Peter Komjach: http://mathoverflow.net/questions/30270/maximum-number-of-mutually-equidistant-points-in-an-n-dimensional-euclidean-spac , geometry - Maximum number of mutually equidistant points in an n-dimensional Euclidean space is (n+1)_ Proof – MathOverflow, answered Jul 2, 2010.

[4]. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений: Пер. с англ. – М.: Мир, 1984. – 333 с.

[5]. Писсанецки С. Технология разреженных матриц: Пер. с англ. – М.: Мир, 1988. – 410 с.

[6]. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ: Под ред. Красикова И. В. — 2-е изд. — М.: Вильямс, 2005. — 1296 с.

Optimal Ordering of Conflicting Objects and the Traveling Salesman Problem

Alexey Voevodin, Semen Kosyachenko
Silver-AVV@yandex.ru , spiero@yandex.ru
Business Center «Video International», Moscow, Russia

Abstract. The paper presents the setting of the problem of optimal ordering of conflicting objects. This problem appears in sociology, in advertising on TV and other media networks. Point is that positions of some conflicting commercials were as far apart as possible if commercials belong to the same class of goods or the same advertiser or brand. Solution algorithms are described for this and related problems such as the Travelling Salesman Problem (TSP). The tasks are considered of two types: when penalized only a juxtaposition of objects according to the TSP, and the generalized case when the penalty acts on the neighboring and on the distant to each other conflicting objects. The penalty depends on the number of objects that lie between them. The TSP with sparse matrix is also considered. For sparse practice cases of the TSP with the band or block diagonal matrix necessary and sufficient conditions are proved to objective function attained its zero minimum and the algorithms guaranteeing exact solution for the TSP are constructed. The proposed algorithms are effective also for sparse matrices of general type. These algorithms can also be successfully used as approximate algorithms for matrices close to the band or block diagonal. For the general case in the paper we propose a heuristic algorithm with high performance and good accuracy of solution. We got good scalability to any size while maintaining linear complexity, which allowed us to use these proposed algorithms in practice for large amounts of data, such as advertising in media networks. The practical results of analytical and numerical investigations of algorithm complexity and solution accuracy are presented.

Keywords: optimal placement; Travelling Salesman Problem; TSP; NP-hard problems; band matrix; sparse matrix; greedy algorithm; penalty function; conflicts, media network; advertising.

References

[1]. Kuzyurin N.N., Fomin S.A. Effektivnye algoritmy i slozhnost’ vychislenij [Efficient algorithms and computational complexity]. Uchebnoe posobie [Tutorial]. Moscow, MIPT’ Publ., 2007. 312 c.
[2]. http://en.wikipedia.org/wiki/Travelling_salesman_problem
[3]. Peter Komjach: http://mathoverflow.net/questions/30270/maximum-number-of-mutually-equidistant-points-in-an-n-dimensional-euclidean-spac, geometry - Maximum number of mutually equidistant points in an n-dimensional Euclidean space is (n+1)_ Proof – MathOverflow, answered Jul 2, 2010.
[4]. Alan George, Joseph W. H. Liu. Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Inc., Englewood Cliffs, N.J. 1981 – 324 pages.
[5]. Sergio Pissanetzky. Sparse Matrix Technology. Academic Press, 1984 – 321 pages.

[6]. Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill. 2009 [1990].