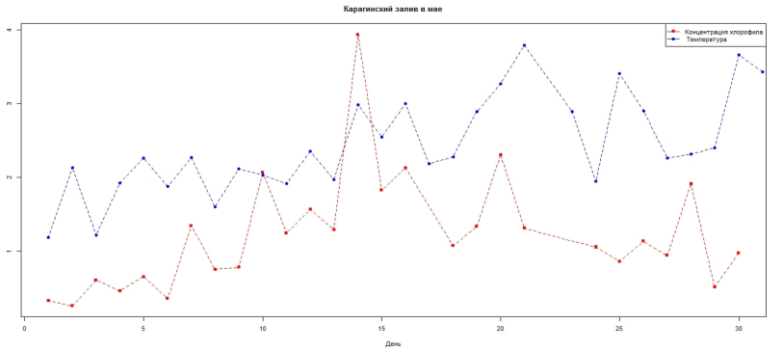

Рисунок 6 – Усреднение спутниковых характеристик по пространству

УДК 004

Практические проблемы реализации алгоритма многомерного анфолдинга для предельно малого множества целей

М.И. Коленко, С.В. Дронов
АлтГУ, г. Барнаул

При работе со статистическими данными нередко возникает проблема их визуализации. При этом данные далеко не всегда имеют числовой характер. Визуализация поможет намного проще воспринимать большие объёмы данных или, к примеру, оценить степень различия изучаемых объектов.

Визуализировать данные просто, если они не более, чем трехмерные. В случае же большего числа измерений одним из самым часто применяемых подходов служит метод перехода от «физических координат» объектов к матрице попарных расстояний или различий между ними. Эта матрица может задаваться и непосредственно, минуя этап многомерного пространства, просто оцениванием попарных расстояний. Далее ставится задача построения изображения скажем, так, чтобы элементы матрицы различий были бы искажены как можно меньше.

Наверное, самым известным методом визуализации в этом случае является метод многомерного шкалирования, см. [1]. При решении задач визуализации модели с неполной матрицей различий исполь-

зуется метод многомерного анфолдинга (PREFSCAL). Поставим эту задачу строго.

Пусть имеется два множества объектов, одно из которых называют множеством целей, а другое – множеством наблюдателей. Расстояния от каждого наблюдателя до каждой цели известны, но при этом попарные расстояния между объектами множеств неизвестны (ни один из наблюдателей не знает расстояние от себя до любого другого наблюдателя, и расстояния между целями также неизвестны). Задача состоит в изображении объектов на плоскости так, чтобы все известные расстояния отображались как можно точнее.

Пусть у нас имеется одна цель и n наблюдателей, расстояния от каждого из наблюдателей до цели известны и обозначаются $R_i, i = 1,...,n$. Поскольку нас интересует только расположение объектов относительно друг друга, мы можем поместить точку, изображающую цель, в начало координат.

Такого рода задача имеет довольно много решений, например, если имеется четыре наблюдателя, при этом все они отдалены на одинаковое расстояние R=5 до цели, то два из возможных решений задачи представлены на рисунке 1. Решений для этих условий имеется бесконечно много.

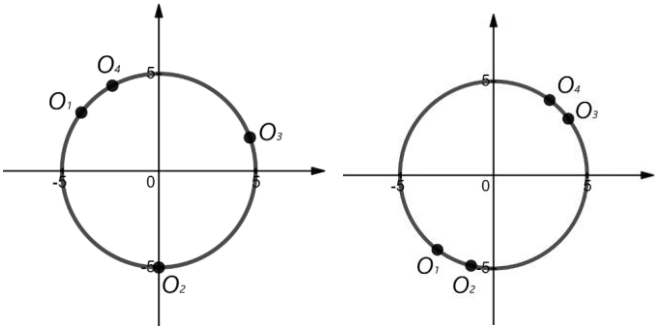

Рисунок 1 – Разные способы построения четырех объектов

В работе [4] было предложено максимизировать минимальные расстояния между объектами. Там же были приведены аргументы в пользу выбора именно такого решения из всего их многообразия.

Алгоритмы решения задачи анфолдинга известны и реализованы в ряде известных статистических компьютерных пакетов [2, 3], но все они работают лишь в предположении, когда и целей, и наблюдателей имеется не менее двух. Авторам известна только одна работа [4], в которой предложен алгоритм для случая единственной цели.

Цель настоящей работы – разработка компьютерной программы, реализующей алгоритм из [4], а также исследование возможностей и характеристик этого алгоритма.

Перед началом упорядочим наблюдателей по возрастанию их расстояний до цели и пронумеруем их соответственно. Единственную цель поместим в начало координат. Без ограничения общности будем считать, что расстояния наблюдателей до цели упорядочены: $R_1 \le R_2 \le ... \le R_n$.

Алгоритм T1-plane

Шаг 1. Расположим точки O_1, O_2 на оси абсцисс, пусть это (R_1, 0), ($-R_2$, 0). Если построение точек не завершено, то к шагу 2. Иначе заканчиваем алгоритм.

Шаг 2. Для построения очередной точки $O_i, i = 3,...,n$ строим окружность радиусом R_i с центром в начале системы координат. К шагу 3.

Шаг 3. Находим минимальную выпуклую оболочку для уже построенных точек. Для этого применим подпрограмму, реализующую алгоритм Джарвиса, взятый из [5]. Описание его дано ниже. На выходе подпрограммы имеем множество точек $S(x_j, y_j),\ j = 1,...,m$ – вершин многоугольника выпуклой оболочки, пронумерованных так, что соседние точки соединяются его стороной. К шагу 4.

Шаг 4. Построим срединные перпендикуляры для каждой стороны минимальной выпуклой оболочки во внешнюю сторону выпуклой оболочки и найдём точки пересечения этих перпендикуляров с окружностью. Для нахождения точек пересечения приходится решать *m* систем с двумя неизвестными, состоящих из линейного и квадратного уравнений. В среде программирования Python 3.7.6 (в которой реализован данный алгоритм), есть библиотека sympy, в которой содержится функция для решения систем уравнений solve_poly_system. Именно эта функция используется для решения системы уравнений, указанной выше.

Пусть получено множество точек $\{C_j,\ j = 1,...,m\}$. К шагу 5.

Шаг 5. Найдём срединный перпендикуляр с максимальной длиной. Поместим точку O_i на место соответствующей этому максимуму точки C_j. Если построение точек не завершено, то к шагу 3. Иначе конец алгоритма.

Опишем алгоритм Джарвиса, используемый в качестве подпрограммы на шаге 3 нашего алгоритма. На вход данной подпрограммы

подаются координаты точек $O(x_i, y_i), i = 1,...,n$. В результате получаются координаты множества вершин минимальной выпуклой оболочки введенных точек.

Шаг 1. Находим самую левую точку среди всего множества точек (то есть точку с минимальной координатой *x*). Добавляем эту вершину в пока что пустое множество *S* вершин минимальной выпуклой оболочки. Называем ее текущей. К шагу 2.

Шаг 2. Для текущей точки (x_i, y_i), выбираем точку $B_j(x_j, y_j)$, для которой, значение $Z = (x_j - x_i)\cdot(y_g - y_j) - (y_j - y_i)\cdot(x_g - x_j) > 0$ для любой из точек $(x_g, y_g), g = 1,...,n, g \neq i, j$. Если точка $B_j(x_j, y_j)$ не совпадает с точкой, найденной на шаге 1, то добавляем ее к *S*, объявляем текущей, и к шагу 2. Если точка $B_j(x_j, y_j)$ равна точке найденной на шаге 1, то алгоритм окончен.

Алгоритм T1 строит каждый раз только одну точку, при этом не изменяя положения остальных относительно друг друга. Но иногда, чтобы оптимально построить новую точку, требуется изменить положение точек уже построенных раньше. Действительно, пусть имеется три наблюдателя, каждый из них сообщает нам одинаковое расстояние до цели *R*=3. Тогда в результате работы алгоритма T1-plane получим рисунок 2. Известно, однако, что три точки на окружности расположены нужным нам образом лишь тогда, когда они лежат в вершинах правильного треугольника. Поэтому алгоритм нуждается в усовершенствовании.

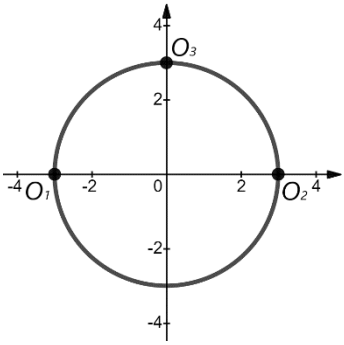

Рисунок 2 – Результат работы T1 для трёх одинаково отдалённых от цели наблюдателей

Алгоритм T1+

Шаг 1. Выберем желаемую точность вычислений ε. Поместим цель в начало координат *O*. Построим точки изображающие наблюдателей при помощи алгоритма T1, назовём полученную конфигурацию точек базовой и обозначим её ***Q***. К шагу 2.

Шаг 2. Найдём попарные расстояния между всеми точками, с первой по последнюю, найдём минимальное из них и обозначим его R_{min}. К шагу 3.

Шаг 3. Пусть R_{min} достигается между точками *A* и *B*. Удалим точку *A*. Разобьем множество остальных на две части – лежащих вне окружности с центром в начале координат и радиусом равным R_A и внутри нее. Для каждой из частей при помощи алгоритма T1 построим потенциально возможные точки для нового положения *A* – всего две точки. Выберем ту из них, для которой длина соответствующего срединного перпендикуляра больше. Найдем в новой конфигурации минимальное расстояние от новой точки до всех остальных. Если оно больше, чем $R_{min} - \varepsilon$, то обозначаем новую конфигурацию ***Q*** и к шагу 5. Иначе вернем новую точку на прежнее место и перейдем к шагу 4.

Шаг 4. Удалим из ***Q*** точку *B*. Построим новое ее положение как на шаге 3. Найдем минимальное расстояние от новой точки до всех остальных. Если оно больше, чем $R_{min} - \varepsilon$, то переобозначим эту комбинацию ***Q***. Иначе вернем точку *B* на прежнее место. К шагу 5.

Шаг 5. Найдём новое минимальное попарное расстояние R_{min2}, если это расстояние меньше, чем $R_{min} - \varepsilon$, или достигается на той же паре точек *A, B*, то возвращаемся к конфигурации ***Q*** и выход из алгоритма. Иначе $R_{min} := R_{min2}$ и возвращаемся к шагу 3.

Для реализации алгоритма T1 была написана программа на языке программирования Python. Она осуществляет построение точек изображающих наблюдателей на плоскости и вывод рисунка на экран.

При запуске программа предложит добавить наблюдателя, получив на это положительный ответ, программа попросит ввести расстояние от наблюдателя до цели. После этого выводятся координаты добавленного наблюдателя и рисунок. При необходимости можно вводить последовательно столько наблюдателей, сколько нужно, и каждый раз будет появляться текущий рисунок. Пример результата работы программы для 5 наблюдателей приведен на рисунке 3.

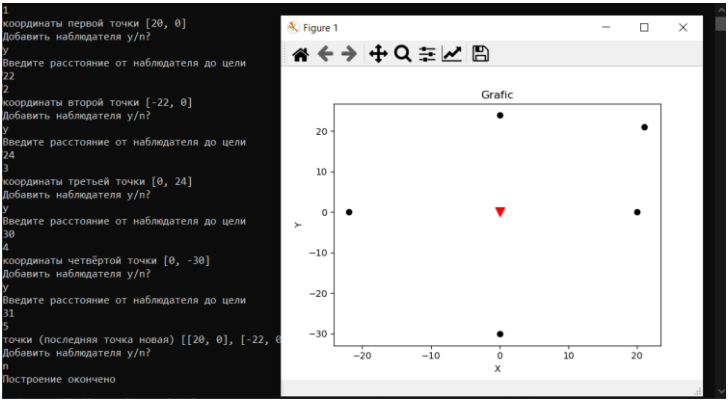

Рисунок 3 – Результат работы программы

После построения всех наблюдателей для работы T1+ требуется задать значение точности ε, после чего начнётся улучшение базовой конфигурации. Закончив передвижение точек, мы получим два рисунка – базовый и оптимальный, а также координаты точек в обоих случаях.

Приведем пример**.** Предположим, что у нас были пять наблюдателей, каждый из которых сообщает одно и то же расстояние до цели R=3. Известно, что максимум минимального расстояния между точками в этом случае достигается для вершин правильного пятиугольника, вписанного в окружность радиуса 3. Выберем ε=0,05. Последовательные этапы работы алгоритма T1+ изображены на рис. 4

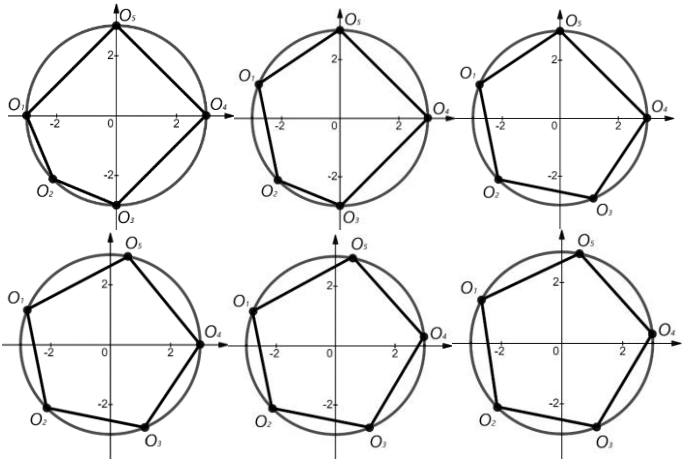

Рисунок 4 – Алгоритм T1+ для пяти точек

Базовая конфигурация имеет вид, изображенный на рисунке 4 слева в верхней части. Для преобразования базовой конфигурации потребовалось 6 итераций, на последней итерации возникла ситуации при которой минимум попарных расстояний возникает на той же паре точек, что и на предыдущей итерации, что позволяет нам закончить алгоритм. В окончательной конфигурации стороны пятиугольника имеют длины $O_1O_2 = 3,5741$, $O_2O_3 = 3,3334$, $O_3O_4 = 3,5742$, $O_4O_5 = 3,5742$, $O_5O_1 = 3,5741$, а длина стороны правильного пятиугольника, вписанного в эту окружность 3,5267, что позволяет сделать вывод о достаточно высоком качестве решения.

По мнению авторов, написанная программа может послужить дополнительным инструментом визуализации многомерных статистических данных в случае неполной информации – когда фактически известны расстояния от каждого из объектов лишь до одного фиксированного объекта. Стандартные статистические пакеты такой возможности пользователю не предоставляют.

Библиографический список

1. Borg I., Groenen P. Modern Multidimensional Scaling: theory and applications (2nd ed.). New York: Springer-Verlag. 2005. 560 p.

2. Компьютерная программа SPSS Statistics. [Электронный ресурс]. – Режим доступа: https://www.ibm.com/analytics/data-science/predictive-analytics/spss-statistical-software. Дата обращения 08.06.2020.

3. STATA – программный пакет для статистического анализа. [Электронный ресурс]. – Режим доступа: https://www.stata.com. Дата обращения 08.06.2020.

4. Dronov S.V., Leongardt K.A. Multidimensional unfolding problem solution in the case of a single target. // IOP Conf. Series: Journal of Physics: Conf. Series 1210. 2019. 012034.

5. Ивановский С.А., Преображенский А.С., Симончик С.К. Алгоритмы вычислительной геометрии. Выпуклые оболочки: простые алгоритмы. // Компьютерные инструменты в образовании. 2007. N1. С. 4 – 19.