Разработка алгоритмов и программных средств решения вычислительных задач идемпотентной алгебры

Голяндина Алина Сергеевна, гр. 522

Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра статистического моделирования

Научный руководитель: д.ф.-м.н., доцент Н.К. Кривулин

Рецензент: к.ф.-м.н., доцент А.Ю. Пономарева

СтатМод

Санкт-Петербург
2011г.

Введение

Дипломная работа содержит:

1. Обзор основных сведений, касающихся идемпотентной алгебры.
2. Формализацию и исследование алгоритмов решения уравнений в $\langle \mathbb{X},\ \oplus,\ \otimes \rangle$:
   - $Ax = b$,
   - $Ax = x$,

   а также неравенства $Ax \leq b$.
3. Программную реализацию.

Общие сведения

$\mathbb{X}$ — числовое множество с операциями сложения $\oplus$ и умножения $\otimes$.
$\langle \mathbb{X}, \ \oplus, \ \otimes \rangle$ — идемпотентное полуполе, то есть

- коммутативное полукольцо с нулем $\mathbb{0}$ и единицей $\mathbb{1}$,
- сложение идемпотентно,
- каждый ненулевой элемент имеет обратный по умножению.

Примеры идемпотентных полуполей

$$\mathbb{R}_{\max,+} = \langle \mathbb{R} \cup \{-\infty\}, \ \max, \ +\rangle, \quad \mathbb{R}_{\min,+} = \langle \mathbb{R} \cup \{+\infty\}, \ \min, \ +\rangle,$$
$$\mathbb{R}_{\max,\times} = \langle \mathbb{R}_+ \cup \{0\}, \ \max, \ \times\rangle, \quad \mathbb{R}_{\min,\times} = \langle \mathbb{R}_+ \cup \{+\infty\}, \ \min, \ \times\rangle,$$

где $\mathbb{R}$ — множество всех вещественных чисел,
$\mathbb{R}_+ = \{x \in \mathbb{R} | x > 0\}$.

Общие сведения

Идемпотентная алгебра матриц

$\mathbb{X}$ — идемпотентное полуполе.

$A = (a_{ij}) \in \mathbb{X}^{m \times n}$ — матрица над полуполем. Операции сложения и умножения матриц, а также операция умножения матрицы на скаляр $x \in \mathbb{X}$ определяются обычным образом:

$$\{A \oplus B\}_{ij} = a_{ij} \oplus b_{ij}, \quad \{BC\}_{ij} = \bigoplus_{k=1}^{n} b_{ik} c_{kj}, \quad \{xA\}_{ij} = x a_{ij}.$$

Обратная матрица здесь существует только для некоторых типов матриц.

Для любой матрицы $A = (a_{ij}) \in \mathbb{X}^{m \times n}$ можно определить псевдообратную матрицу $A^{-} = (a_{ij}^{-}) \in \mathbb{X}^{n \times m}$ с элементами

$$a_{ij}^{-} = \begin{cases} a_{ji}^{-1}, & \text{если } a_{ji} \neq \mathbb{0}, \\ \mathbb{0}, & \text{если } a_{ji} = \mathbb{0}. \end{cases}$$

Линейные уравнения первого рода

Существование и единственность решения

Для $A \in \mathbb{X}^{m \times n}$, $b \in \mathbb{X}^{m}$ **линейным уравнением первого рода** относительно неизвестного вектора $x \in \mathbb{X}^{n}$ называется уравнение

$$Ax = b.$$

Определим величину

$$\Delta(A,\ b) = (A(b^{-}A)^{-})^{-}b.$$

Теорема (Кривулин, 2009)

Уравнение $Ax = b$ имеет решение тогда и только тогда, когда $\Delta(A,\ b) = \mathbb{1}$.
При этом $x = (b^{-}A)^{-}$ является максимальным решением. Если столбцы матрицы A образуют минимальную систему векторов, порождающую b, то других решений нет.

Линейные уравнения первого рода

Общее решение уравнения

$A = (a_1, a_2, \dots, a_n),\ a_i \in \mathrm{X}^m$.
$\mathcal{I} = \big\{ I \mid b \in \mathrm{span}\{a_i | i \in I\}$ и $\forall I' \subset I\ b \notin \mathrm{span}\{a_i | i \in I'\} \big\}$.
Тогда для $\forall I \in \mathcal{I}$ $\{a_i \mid i \in I\}$ — минимальная порождающая b система векторов.

Определим диагональную матрицу

$$G_I = \mathrm{diag}(g_1(I),\ \dots,\ g_n(I)),$$

где $g_i(I) = \mathbb{0}$, если $i \in I$, и $g_i(I) = \mathbb{1}$, если $i \notin I$.

Теорема (Кривулин, 2009)

Пусть уравнение $Ax = b$ разрешимо. Тогда его общим решением является семейство решений

$$x_I = (b^- A \oplus v^T G_I)^-,\ v \in \mathbb{X}^n,\ I \in \mathcal{I}.$$

Формализация алгоритмов

Основная сложность при решении уравнения $Ax = b$ — поиск всех $I \in \mathcal{I}$, где
$\mathcal{I} = \{I \mid b \in \operatorname{span}\{a_i | i \in I\}$ и $\forall I' \subset I \; b \notin \operatorname{span}\{a_i | i \in I'\}\}$
задает множество всех минимальных порождающих систем.

Полный перебор — очень трудоемкая процедура.

Проблема решается с помощью построения дерева, вершины которого соответствуют наборам индексов столбцов.

Рассмотрим 3 алгоритма построения такого дерева.

Описание алгоритмов

Алгоритм №1 Строим дерево, каждой вершине соответствует набор индексов I. Для корневой вершины $I = \{1,\ 2,\ \ldots,\ n\}$.

- Для каждой вершины сначала проверяется условие Теоремы о существовании и единственности решения для подмножества столбцов матрицы A.
- Если условие выполняется, то к дереву присоединяются вершины, соответствующие наборам индексов $I' = I \backslash \{i\}$ для каждого $i \in I$.
- Иначе текущая вершина считается концевой и далее не рассматривается.

Процедура завершается, когда все вновь присоединенные вершины оказываются концевыми. Искомым наборам индексов отвечают вершины, у которых все присоединенные вершины — концевые. Повторяющиеся наборы удалим.

Алгоритм №1

Пример:

$n = 4$, $\mathcal{I} = \{\{1, 2, 3\}, \{1, 4\}\}$. Представим I при помощи вектора $v(i)$, элемент которого $v_j(I)$ — индикатор $j \in I$.

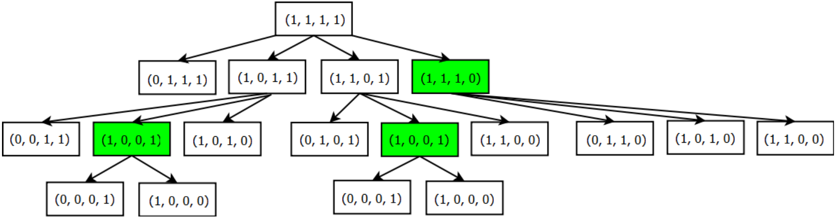

Рис.: Дерево, построенное в соответствии с Алгоритмом №1

Описание алгоритмов

Алгоритм №2

Модификация Алгоритма №1: отсеиваем повторяющиеся наборы индексов в процессе построения дерева.

Перед тем, как добавлять вершину в дерево, необходимо проверить, нет ли в построенном на данный момент дереве аналогичной вершины.

Заключительный шаг: отсеять избыточные наборы.

Таким образом мы существенно уменьшим размер дерева и, соответственно, количество проверок на разрешимость, однако увеличим количество сравнений.

Алгоритм №2

Пример:

$n = 4$, $\mathcal{I} = \{\{1, 2, 3\}, \{1, 4\}\}$. Закодируем I вектором $v(I): \; v(I)_j \neq 0 \Leftrightarrow j \in I$.

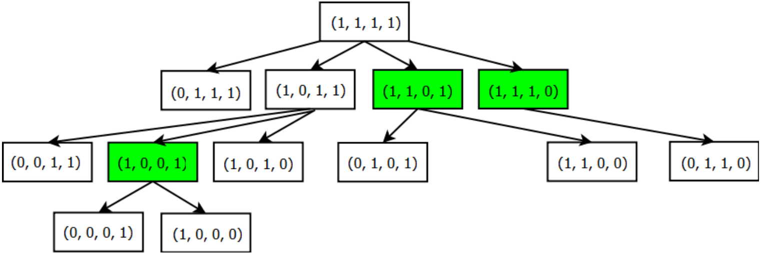

Рис.: Дерево, построенное в соответствии с Алгоритмом №2

Линейные уравнения первого рода

Алгоритм №3

- Для каждой вершины проверяем условие Теоремы о существовании и единственности решения для подмножества столбцов матрицы A.
- Обозначим через j индекс, вычеркивание которого привело в данную вершину. Для корневой вершины $j = 0$. Если условие из пункта 1 выполняется, то к дереву присоединяются новые вершины, соответствующие наборам $I' = I \backslash \{i\}$ для каждого $i \in I$, такого что $i > j$.
- Иначе текущая вершина удаляется из дерева, а для исследования выбирается следующая.

Рассмотрим наборы индексов, соответствующие концевым вершинам. Они содержат как искомые подмножества, так и более широкие. Необходимо удалить избыточные.

Алгоритм №3

Пример:

$n = 4$, $\mathcal{I} = \{\{1, 2, 3\}, \{1, 4\}\}$. Закодируем I вектором $v(I): \ v(I)_j \neq 0 \Leftrightarrow j \in I$.

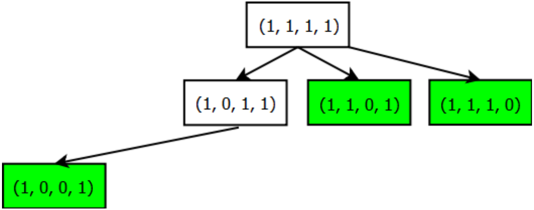

Рис.: Дерево, построенное в соответствии с Алгоритмом №3

Сравнение алгоритмов

Основная трудоемкость заключается в вычислении
$\Delta = (A(b^- A)^-)^- b$.
По построению видно, что для 2-го и 3-го алгоритмов всегда требуется меньше вычислений Δ, чем для 1-го.

Сравнение алгоритмов на примере

- Алгоритм 1: 18 вычислений Δ,
  3 сравнения векторов размера $n = 4$.
- Алгоритм 2: 13 вычислений Δ, 81 сравнение.
- Алгоритм 3: 9 вычислений Δ, 3 сравнения.

Линейные уравнения первого рода

Сравнение алгоритмов

С увеличением высоты дерева различие между алгоритмами становится более существенным.

С помощью моделирования систем уравнений выяснено

- Для матриц $A^{m \times n}$, у которых $m < n$, средняя высота дерева неограниченно растет с ростом числа столбцов.
- Для остальных матриц средняя высота дерева меньше 2.

Следовательно, для матриц, у которых $m < n$ особенно важно выбрать наиболее эффективный алгоритм. А для случая $m \geq n$ можно воспользоваться любым из алгоритмов.

Линейные уравнения второго рода

Необходимые определения

Однородное уравнение второго рода: $Ax = x$.

Матрица называется разложимой, если при помощи перестановки строк вместе с такой же перестановкой столбцов ей может быть придана нормальная форма.

Нормальная форма разложимой матрицы:

$$A = \begin{pmatrix} A_{11} & \mathbb{0} & \dots & \mathbb{0} \\ A_{21} & A_{22} & & \mathbb{0} \\ \vdots & \vdots & \ddots & \\ A_{s1} & A_{s2} & \dots & A_{ss} \end{pmatrix},$$

где A_{ii} — неразложимая или нулевая квадратная матрица порядка n_i, A_{ij} — произвольная матрица размера $n_i \times n_j$ для всех $j < i$, $i = 1, \dots, s$, при условии $n_1 + \dots + n_s = n$.

Алгоритм решения

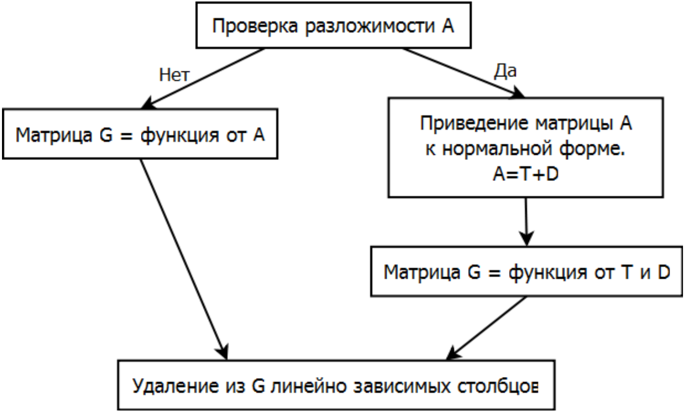

Рис.: Краткая схема алгоритма решения однородного уравнения второго рода. Матрица G задает пространство решений уравнения.

Программная реализация

Общее описание

Разработаны программные средства для решения обобщенных линейных уравнений над произвольным полуполем.
В частности, $\mathbb{R}_{\max,+}$, $\mathbb{R}_{\min,+}$, $\mathbb{R}_{\max,\times}$, $\mathbb{R}_{\min,\times}$.
Программа позволяет найти

- как максимальное, так и общее решение уравнения первого рода $Ax = b$,
- общее решение однородного уравнения второго рода $Ax = x$.

Программная реализация состоит из

- тестового модуля (1 файл),
- библиотеки классов (5 файлов, 9 классов).

Среда разработки — Microsoft Visual Studio 2008. Язык разработки — C++ с использованием шаблонов.

Программная реализация

Структура

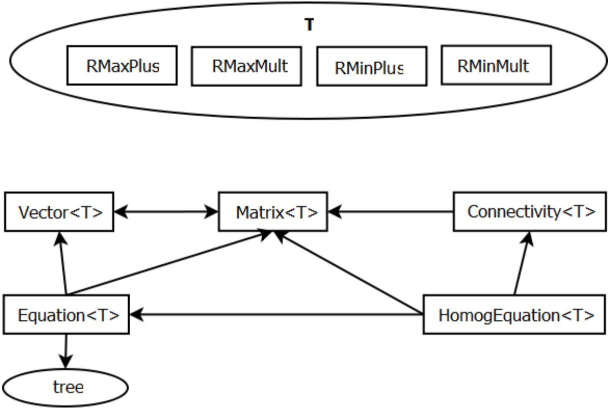

Рис.: Диаграмма использования классов

Программная реализация

Сравнение с существующими пакетами

Среди существующих программных средств не найдено эквивалентных по возможностям.

Примеры существующих:

- MAXPLUS2.3.3 MaxPlus Toolbox of Scilab — решение уравнение $Ax = b$ не рассматривается,
- Max-Plus Algebra Toolbox — для уравнения $Ax = b$ осуществляется поиск только максимального решения.

Все рассмотренные программные средства ориентированы на решение задач только для одного полуполя (обычно $\mathbb{R}_{\max,+}$ или $\mathbb{R}_{\min,+}$).

Итоги

- Рассмотрены 3 алгоритма решения уравнения $Ax = b$.
- Проведено сравнение алгоритмов.
- Так как трудоемкость определяется высотой дерева, с помощью моделирования оценена средняя высота дерева для разных форм матрицы $A^{m \times n}$. Вывод: выбор эффективного алгоритма особенно важен при $n > m$.
- Проведена формализация алгоритмов решения $Ax = x$.
- Выполнена программная реализация решения уравнений вида $Ax = b$, $Ax = x$ и неравенства $Ax \leq b$ над $\langle \mathbb{X}, \oplus, \otimes \rangle$.
- Программная реализация вычислительных процедур является универсальной и не зависит от конкретного полуполя.