Вычислительные технологии, 2020, том 25, № 6, с. 85–94. © ФИЦ ИВТ, 2020
Computational Technologies, 2020, vol. 25, no. 6, pp. 85–94. © FRC ICT, 2020
ISSN 1560-7534
eISSN 2313-691X

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

DOI:10.25743/ICT.2020.25.6.005

Построение высокоэффективных распределенных систем автоматизации схемотехнического проектирования на основе сжатия данных

В. И. Анисимов[1,2], В. Н. Гридин[1,*]
[1]Центр информационных технологий в проектировании РАН, 143003, Одинцово, Россия
[2]Санкт-Петербургский электротехнический университет "ЛЭТИ", 197022, Санкт-Петербург, Россия
*Контактный автор: Гридин Владимир Николаевич, e-mail: info@ditc.ras.ru

Поступила 7 октября 2020 г., доработана 11 декабря 2020 г., принята в печать 16 декабря 2020 г.

Рассмотрены вопросы повышения эффективности работы распределенных систем автоматизации схемотехнического проектирования путем минимизации времени взаимодействия веб-сервисов с сетью Интернет на основе методов сжатия данных. Приведены результаты сравнительной оценки эффективности основных методов компактной обработки разреженных матриц при использовании для сжатия данных метода индексно-адресных матриц, списковых схем хранения, метода строчного фиксированного формата, метода строчно-столбцового формата. Известной особенностью методов фиксированного формата является невозможность включения в сжатое описание новых ненулевых элементов, появляющихся при решении уравнений моделируемой схемы. На основе сравнительного анализа методов сжатия данных показано, что наибольший интерес для практической реализации высокоэффективного программного обеспечения систем автоматизированного проектирования представляет метод списковых схем, имеющий наибольшие возможности для сканирования матриц в произвольном направлении. Существенное достоинство метода списковых схем — возможность включения в компактное описание дополнительных ненулевых элементов, которые образуются при решении уравнений.

Ключевые слова: системы автоматизированного проектирования, веб-сервисы, моделирование систем, компактная обработка, разреженные матрицы, распределенные системы, Интернет-технологии.

Цитирование: Анисимов В.И., Гридин В.Н. Построение высокоэффективных распределенных систем автоматизации схемотехнического проектирования на основе сжатия данных. Вычислительные технологии. 2020; 25(6):85–94. DOI:10.25743/ICT.2020.25.6.005.

Введение

При организации работы распределенных систем автоматизированного проектирования особенно актуальны вопросы увеличения их производительности, поскольку для повышения эффективности функционирования распределенной системы необходимо минимизировать время взаимодействия веб-сервисов с сетью Интернет [1–6]. Такая задача может быть решена путем перехода к компактной форме хранения и обработки разреженных матриц на основе тех или иных методов сжатия данных [7–10].

Наличие в математическом описании моделируемых систем разреженных матриц ставит задачу изменения стандартных подходов к формированию и решению систем уравнений. Это объясняется, с одной стороны, требованием экономии памяти, которую нежелательно использовать для хранения нулевых элементов, а с другой — требованием повышения быстродействия за счет устранения выполнения арифметических операций с нулевыми элементами. Применяемый обычно для этого способ логической проверки элементов с целью устранения арифметических операций с нулевыми элементами не приводит к ожидаемому эффекту увеличения быстродействия программного обеспечения, поскольку для выполнения логических операций проверки элементов необходимо затратить определенное время.

Степень разреженности матриц оценивается коэффициентом разреженности

$$\alpha = \frac{m}{m_{\max}} = \frac{m}{n^2},$$

где m — число ненулевых элементов матрицы, n — порядок матрицы.

Известные методы компактной обработки разреженных матриц существенно различаются по своим характеристикам и своей эффективностью, поэтому для правильного выбора соответствующего метода необходимо провести их сравнительную оценку, которая приводится ниже для всех возможных методов сжатия данных.

1. Сравнительная оценка эффективности методов сжатия данных

Связные схемы хранения. К связным схемам относятся классический метод Кнута и сокращенные варианты основного метода [11–14]. При использовании основного метода Кнута необходимо создать следующие массивы:

WZ — для значений ненулевых элементов w_{ij} исходной матрицы;

WI — для номеров строк ненулевых элементов;

WJ — для номеров столбцов ненулевых элементов;

NR — для хранения относительного адреса α следующего ненулевого элемента строки (α — порядковый номер элемента в массиве WZ);

NC — для хранения относительного адреса α следующего ненулевого элемента столбца;

ER — для относительного адреса α входа в очередную строку;

EC — для относительного адреса α входа в очередной столбец.

Весьма важной особенностью метода Кнута является возможность записи элементов массива WZ в любом порядке и, как следствие, введения в описание дополнительных ненулевых элементов. Достоинство метода Кнута — возможность сканирования исходной матрицы как по строкам, так и по столбцам, некоторым недостатком метода является значительное число массивов, необходимых для хранения информации о ненулевых элементах исходной матрицы.

Если учесть, что массивы WZ, WI, WJ, NR, NC имеют длину m, определяемую числом ненулевых элементов, а массивы ER, EC имеют длину n, определяемую порядком исходной матрицы, то коэффициент сжатия данных для метода Кнута может быть найден с использованием выражения

$$\beta = \frac{8 \cdot n^2}{16 \cdot n^2 \cdot \alpha + 4 \cdot n} \approxeq \frac{1}{2 \cdot \alpha}.$$

Из полученного выражения видно, что с уменьшением α эффективность метода неограниченно возрастает.

Сокращенный метод Кнута позволяет уменьшить число используемых массивов путем исключения из полного описания массивов, которые реализуют сканирование по столбцам. При этом для компактного описания исходной матрицы необходимо ввести только массивы WZ, WJ, NR, ER. Аналогично можно составить вариант сокращенной схемы Кнута, позволяющий осуществить сканирование только по столбцам. Для компактного описания необходимо использовать только массивы WZ, EC, NC, WI.

Коэффициент сжатия данных любой из схем сокращенного метода Кнута может быть определен выражением

$$\beta = \frac{8 \cdot n^2}{n^2 \cdot \alpha \cdot 12 + n^2} \approxeq \frac{1}{1.5 \cdot \alpha}.$$

Очевидно, что эффективность сокращенной схемы Кнута выше, чем эффективность полной схемы Кнута, однако это достигается за счет ограничения возможности выбора направления сканирования, что является некоторым недостатком сокращенного метода Кнута. Однако возможность введения в описание дополнительных ненулевых элементов, а также произвольный порядок их записи в массиве остаются в силе. Это обстоятельство можно считать существенным достоинством сокращенного метода Кнута.

Методы фиксированного формата. К методам фиксированного формата относятся методы строчного и строчно-столбцового формата.

Для использования метода строчного фиксированного формата требуются следующие массивы:

WZ — для хранения значения ненулевых элементов w_{ij} исходной матрицы;

WJ — для хранения индексов столбцов ненулевых элементов исходной матрицы W;

ER — массив, содержащий указатели точек входа в очередную строку.

Длина массивов WZ, WJ составит m элементов, а длина массива ER — $n+1$ элементов, при этом в $n+1$ заносится значение $m+1$.

Отличительными особенностями метода являются невозможность произвольного выбора порядка записи ненулевых элементов в массиве WZ и, как следствие, невозможность включения в описание дополнительных ненулевых элементов. Это объясняется тем, что формат всех массивов жестко зафиксирован и не может меняться произвольным образом в процессе расчета.

Коэффициент сжатия данных метода строчного фиксированного формата может быть определен выражением

$$\beta = \frac{8 \cdot n^2}{n^2 \cdot \alpha \cdot 10(n+1) \cdot 2} \approxeq \frac{1}{1.25 \cdot \alpha}.$$

Очевидно, что эффективность этого метода выше, чем эффективность рассмотренных ранее. Существенный его недостаток — невозможность включения в описание дополнительных элементов, что объясняется наличием фиксированного формата.

Метод строчно-столбцового формата также основан на использовании фиксированного формата, однако, в отличие от предыдущего метода, предполагается, что исходная матрица является структурно-симметричной, так что для каждого ненулевого элемента w_{ij} можно поставить в соответствие элемент w_{ji}. В случае, если такой элемент в исходной матрице отсутствует, то его необходимо создать искусственно путем включения в компактное описание элемента $w_{ji} = 0$.

Для компактного хранения исходных элементов матрицы создают три массива:

WD — для хранения диагональных элементов;

WL — для хранения ненулевых элементов, расположенных ниже диагонали (поддиагональных элементов);

WU — для хранения ненулевых элементов, расположенных выше диагонали (наддиагональных элементов).

В соответствии с методом при формировании массива WU наддиагональные элементы записываются по строкам, а при формировании массива WL поддиагональные элементы — по столбцам. Согласно принятому порядку формирования массивов WU, WL, относительный адрес некоторого элемента w_{ij}, расположенного в массиве WU, совпадает с относительным адресом элемента w_{ji}, расположенного в массиве WL. Отмеченное свойство существенно упрощает процесс программирования для организации сканирования элемента.

Для хранения индексов строк и столбцов ненулевых элементов в методе строчно-столбцового фиксированного формата используется массив WJI. Этот массив содержит номера столбцов ненулевых элементов, расположенных выше диагонали, которые совпадают с номерами строк транспонированных ненулевых элементов, расположенных ниже диагонали. Для определения точки входа в строку выше диагонали (точка входа в столбце ниже диагонали) используется массив ERC. В последний n-й элемент этого массива заносится значение $\dfrac{m-n}{2}+1$.

Если учесть, что длина массива WD составляет n элементов, длина массивов WU, WL, WJI составляет $\dfrac{m-n}{2}$ элементов, а длина массива ERC составляет n элементов, то коэффициент сжатия данных для метода строчно-столбцового фиксированного формата может быть определен выражением

$$\beta = \frac{8 \cdot n^2}{n^2 \cdot \alpha \cdot 9 + n} \approx \frac{8}{9} = \frac{1}{1.1 \cdot \alpha}.$$

Отсюда следует, что эффективность этого метода выше, чем всех рассмотренных ранее методов компактной обработки разреженных матриц. Однако, так же как для метода строчного фиксированного формата, метод строчно-столбцового фиксированного формата в его классической форме не позволяет перечислять ненулевые элементы в произвольном порядке, что не позволяет включать дополнительные элементы в компактное описание.

Метод индексно-адресных матриц. Метод основан на введении некоторой целочисленной матрицы. Эта матрица в точности повторяет структуру исходной матрицы W, входящей в уравнение моделируемой системы. В качестве элементов индексно-адресная матрица содержит порядковый номер α ненулевых элементов матрицы W, которые перечисляются в некотором массиве WZ. Если порядковый номер ненулевого элемента исходной матрицы равен α, то в индексно-адресную матрицу вводится значение $A(i,j) = \alpha$.

При практической реализации метода индексно-адресных матриц достаточно провести сканирование индексно-адресной матрицы и выбрать из нее очередной ненулевой элемент. Затем, учитывая его численное значение, необходимо выбрать соответствующий порядковый элемент из массива WZ. Коэффициент сжатия данных для метода индексно-адресных матриц в случае, когда для хранения индексно-адресной матрицы A используется тип данных длиной два байта, а для хранения каждого значащего элемен-

та исходной матрицы используются данные длиной 8 байтов, может быть определен выражением

$$\beta = \frac{8n^2}{n^2(\alpha \cdot 8 + 2)} = \frac{1}{\alpha + 0.25}.$$

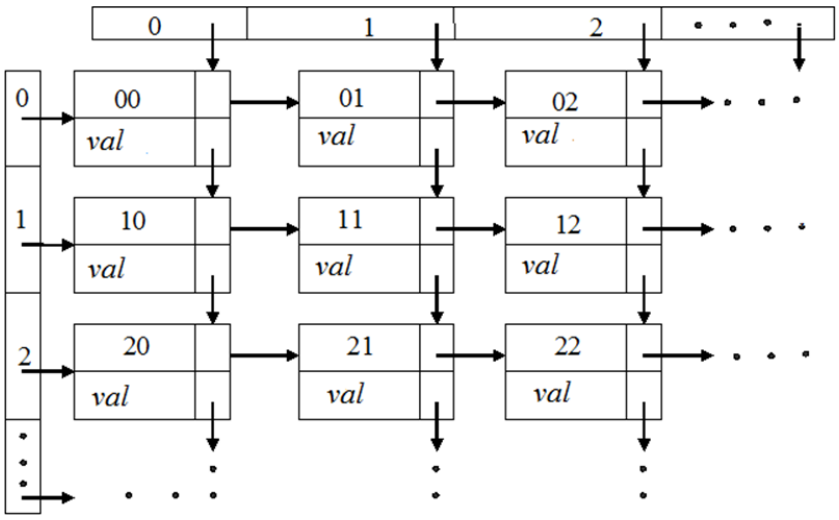
Из приведенного соотношения видно, что коэффициент сжатия данных для рассмотренного метода всегда меньше 4. Следовательно, метод индексно-адресных матриц характеризуется незначительной эффективностью при компактной записи разреженных матриц.

На основании сравнительной оценки методов сжатия данных следует, что наибольшую эффективность имеет метод строчно-столбцового фиксированного формата. Однако, поскольку формат всех массивов этого метода жестко зафиксирован и не может меняться произвольным образом в процессе расчета, его отличительной особенностью является невозможность включения в описание дополнительных ненулевых элементов. Это обстоятельство исключает возможность непосредственного применения строчно-столбцового фиксированного формата для обработки информации при решении систем уравнений любым численным методом вследствие неизбежного появления новых ненулевых элементов в процессе этого решения. Указанный недостаток отсутствует в методе Кнута, который имеет довольно высокую эффективность и позволяет осуществить запись элементов сжатого массива в любом порядке и, как следствие, предоставляет возможность введения в описание дополнительных ненулевых элементов. Достоинством метода Кнута является также возможность сканирования исходной матрицы как по строкам, так и по столбцам. В ряде случаев может оказаться целесообразным использование сокращенных методов Кнута, которые имеют более высокую эффективность.

2. Программная реализация связных схем обработки данных

Практическая реализация методов Кнута связана с необходимостью полной перекодировки всех массивов при введении в описание дополнительных элементов, что связано с большими техническими трудностями в процессе построения программного обеспечения и снижением производительности его дальнейшего функционирования. Возникающие проблемы могут быть устранены путем перехода к объектно-ориентированному подходу и описанию схемы Кнута на основе существующей в современных языках программирования C# и Java методики работы с коллекциями объектов класса *ArrayList*, предназначенного для поддержки динамических массивов, размеры которых могут изменяться в процессе работы [15–18]. Класс *ArrayList* реализует ряд интерфейсов и помимо свойств и методов, определенных в интерфейсах, имеет также и собственные свойства и методы.

При создании списка, обеспечивающего работу с некоторыми разреженными матрицами, в общем случае необходимо реализовать обработку этой матрицы с выполнением сканирования как по строкам, так и по столбцам. Для решения такой обобщенной задачи необходимо создать два объекта *rows* и *columns* класса *ArrayList*. Каждый элемент объекта *rows* соответствует определенной строке матрицы и либо ссылается на ненулевой элемент в данной строке, либо указывает в *NULL*, если в строке элементов нет. Каждый элемент объекта *columns* соответствует определенному столбцу матрицы и либо ссылается на ненулевой элемент в данном столбце, либо указывает в *NULL*, если в столбце элементов нет.



Связь списков с объектами *rows* и *columns*
Associating lists with *row* and *column* objects

Для описания элементов в списке следует создать класс *Element*, в качестве членов которого используются переменные *double val* для задания значения элементов исходной матрицы и переменные *int i*, *int j* для задания индексов строк и столбцов этих значений. Кроме того, членами класса являются ссылки типа *Element nextI* и *Element nextJ* для указания на следующий элемент в строке и столбце. Методы класса включают в себя единственый конструктор с тремя аргументами для создания экзампляра класса. Общая структура, отображающая связь списков с объектами *rows* и *columns*, приведена на рисунке.

Для описания всей разреженной матрицы следует ввести класс *Matrix*, членами которого являются ссылка *Element el*, а также объекты *rows* и *columns*. Методы класса включают в себя методы *addToRows*, *addToColumns*, *resize* и *setAt*, посредством которых реализуется добавление новых элементов, и методы *getAtRows*, *getAtColumns*, которые обеспечивают доступ к элементам строк и столбцов. Метод *setAt* предназначен для формирования элементов списка. Работа метода *setAt* начинается с вызова метода *resize*. Этот метод предназначен для увеличения размеров объектов класса коллекций *rows* и *columns* в том случае, когда новый элемент имеет значение координат, большее текущих для этих объектов. Значения текущих элементов задаются свойством *Count* класса *ArrayList*. Методы *getAtRows* и *getAtColumns* предназначены для получения информации о содержимом рядов и столбцов, структура этих методов имеет идентичный характер.

Реализация методов включения в список новых элементов и методов для доступа к элементам списка базируется на использовании стандартных методов, определенных в интерфейсах класса *ArrayList*.

Алгоритм добавления в список элемента с координатами i, j и вещественным значением *val* заключается в следующем.

1. Из списка строк объекта выбирается ссылка с номером строки i.
2. Если текущая ссылка нулевая, то она настраивается на новый элемент с координатами (i, j) и значением *val*.
3. Если координата j элемента по текущей ссылке равна координате j нового элемента, то у него заменяется значение *val*.

4. Если координата j элемента по текущей ссылке больше, чем координата j нового элемента, то новый элемент должен быть вставлен в начало списка i-й строки. Ссылка на следующий у нового элемента настраивается на элемент по текущей ссылке.
5. Вводятся понятия ссылки на предыдущий, равной текущей ссылке, и ссылки на следующий, равной ссылке на следующий у элемента по текущей ссылке.
6. Если ссылка на следующий нулевая, переходим к шагу 10.
7. Если координата j элемента по ссылке на следующий равна координате j нового элемента, то элемент с такими координатами в матрице уже есть и у него заменяется значение *val*.
8. Если координата j объекта по ссылке на следующий больше, чем координата j нового элемента, то новый элемент должен быть вставлен между предыдущим и следующим. Ссылка на следующий у предыдущего элемента настраивается на новый созданный элемент, а ссылка на следующий у нового элемента приравнивается к ссылке на следующий у предыдущего элемента.
9. Перебираются элементы по ссылкам на предыдущий и на следующий; каждая ссылка настраивается на ссылку на следующий для рассматриваемого объекта. Переходим к шагу 6.
10. Если за время работы алгоритма ни один элемент не был изменен или вставлен, то ссылка на следующий у элемента по ссылке на предыдущий настраивается на новый созданный элемент, т. е. элемент становится последним в строке.

Эффективность списковой схемы при использовании коллекций по сравнению с классической схемой Кнута несколько снижается, так как для каждого объекта класса *Element* помимо значений *val*, i, j необходимо также хранить значения ссылок *nextI* и *nextJ*.

Заключение

Рассмотрение методов сжатия данных на основе компактной обработки разреженных матриц позволяет сделать вывод, что наибольший интерес для практической реализации высокоэффективного программного обеспечения систем автоматизированного проектирования представляет метод списковых схем, имеющий наибольшие возможности для сканирования матриц в произвольном направлении. Важным достоинством метода является возможность включения в компактное описание дополнительных ненулевых элементов, неизбежно появляющихся в процессе решения уравнений. Практическая реализация методов компактной обработки данных позволяет существенно увеличить производительность работы веб-сервисов распределенных систем автоматизированного проектирования и повысить надежность работы системы вследствие уменьшения времени взаимодействия с сервером в процессе эксплуатации САПР.

Благодарности. Работа выполняется в рамках темы № 0071-2019-0001.

Список литературы

[1] **Анисимов В.И., Гридин В.Н.** Методы построения систем автоматизированного проектирования на основе Интернет-технологий и компактной обработки разреженных матриц. Информационные технологии в проектировании и производстве. 2009. (1):3–7.

[2] **Коваленко О.С., Курейчик В.М.** Обзор проблем и состояний облачных вычислений и сервисов. Известия ЮФУ. Технические науки. 2012; 7(132):146–153.

[3] **Гридин В.Н., Анисимов В.И., Ахмад А.Д.** Построение клиентских .NET-приложений в распределенных схемотехнических системах автоматизированного проектирования. Системы и средства информатики. 2016; 26(1):76–85. DOI:10.14357/08696527160106.

[4] **Гридин В.Н., Дмитревич Г.Д., Анисимов Д.А.** Построение систем автоматизированного проектирования на основе web-технологий. Информационные технологии. 2011. (5):23–26.

[5] **Ларистов Д.А., Анисимов Д.А.** Построение встроенного web-интерфейса в системах автоматизации проектирования. Известия СПбГЭТУ "ЛЭТИ". Сер.: Информатика, управление и компьютерные технологии. 2007. (2):63–66.

[6] **Анисимов Д.А.** Методы построения систем автоматизации схемотехнического проектирования на основе веб-сервисов. Известия СПбГЭТУ. 2012. (10):56–61.

[7] **Писсанецки С.** Технология разреженных матриц. М.: Мир; 1988: 410. ISBN:5-03-000960-4.

[8] **Эстербю О., Златев З.** Прямые методы для разреженных матриц. М.: Мир; 1987: 118.

[9] **Норенков И.П.** Введение в автоматизированное проектирование технических устройств и систем. М.: Высшая школа; 1986: 302.

[10] **Гридин В.Н., Анисимов В.И., Абухазим М.М.** Сжатие данных в системах автоматизации схемотехнического проектирования на основе методов фиксированного формата. Системы высокой доступности. 2016. (4):34–40.

[11] **Кнут Д.** Искусство программирования для ЭВМ. Т. 1. М.: Мир; 1976: 734.

[12] **Влах И., Сингхал К.** Машинные методы анализа и проектирования электронных схем. М.: Радио и связь; 1988: 560.

[13] **Анисимов В.И., Амахвр Ю.М.** Компактные методы обработки разреженных матриц задач мониторинга на основе списочных структур. Труды Пятой Международной конференции "Приборостроение в экологии и безопасности человека". СПб.: ГУАП; 2007: 157–160.

[14] **Амахвр Ю.М.** Компактная обработка разреженных матриц на основе списков. Вестник компьютерных и информационных технологий. 2008. (4):49–52.

[15] **Дейтел Х.М., Дейтел П.Дж.** Как программировать на Java: Введение в объектно-ориентированное проектирование с использованием UML-образцов проектирования JAVA / Пер. с англ. под ред. А.В. Козлова. Кн. 1. М.: Бином пресс; 2003: 847.

[16] **Троелсен Э.** Язык программирования C# 2005 и платформа .NET 2.0. 3-е изд. М.: Вильямс; 2007: 1154. ISBN:5-8459-1124-9.

[17] **Шилдт Г.** Полный справочник по C#. М.: Вильямс; 2004: 752. ISBN:5-8459-0563-X.

[18] **Мак-Дональд М., Шпушта М.** Microsoft ASP.NET 2.0 с примерами на C# 2005 для профессионалов. М.: Вильямс; 2006: 1408. ISBN:5-8459-1091-9.

Вычислительные технологии, 2020, том 25, № 6, с. 85–94. © ФИЦ ИВТ, 2020 ISSN 1560-7534
Computational Technologies, 2020, vol. 25, no. 6, pp. 85–94. © FRC ICT, 2020 eISSN 2313-691X

INFORMATION TECHNOLOGIES

DOI:10.25743/ICT.2020.25.6.005

Building highly efficient distributed automation systems for circuit design based on data compression

Anisimov Vladimir I.[1,2], Gridin Vladimir N.[1,*]

[1]Design Information Technologies Center Russian Academy of Sciences, 143003, Odintsovo, Russia
[2]Saint Petersburg Electrotechnical University "LETI", 197022, Saint Petersburg, Russia
[*]Corresponding author: Gridin Vladimir N., e-mail: info@ditc.ras.ru
Received October 7, 2020, revised December 11, 2020, accepted December 16, 2020

Abstract

This article considers the issues of increasing the efficiency of distributed automation systems for circuit design by minimizing the interaction time of web services with the Internet based on data compression methods. The results of a comparative assessment of the effectiveness for the main methods of compact processing of sparse matrices using the method of index-address matrices, list storage schemes, the method of row fixed format, and the method of row-column format are presented. A well-known feature of the fixed-format methods is the impossibility of including in the concise description of new nonzero elements that appear when solving the equations of the simulated circuit. Comparative analysis of data compression methods shows that the method of list circuits, which provides the greatest opportunities for scanning matrices in an arbitrary direction, is of great interest for implementation of highly efficient software for computer-aided design systems. An essential advantage of the list circuit method is the possibility of including additional nonzero elements in the compact description, which are formed when solving equations.

Keywords: computer-aided design systems, web services, systems modelling, compact processing, sparse matrices, distributed systems, Internet technologies.

Citation: Anisimov V.I., Gridin V.N. Building highly efficient distributed automation systems for circuit design based on data compression. Computational Technologies. 2020; 25(6):85–94. DOI:10.25743/ICT.2020.25.6.005. (In Russ.)

Acknowledgements. The work is carried out within the framework of theme No. 0071-2019-0001.

References

1. **Anisimov V.I., Gridin V.N.** Methods of construction of systems of the automated designing on the basis of the internet-technologies and compact processing of the rarefied matrixes. Informatsionnye Tekhnologii v Proektirovanii i Proizvodstve. 2009; (1):3–7. (In Russ.)
2. **Kovalenko O.S., Kureichik V.M.** Review of problems and aspects about cloud computing and services. Izvestiya SFedU. Engineering Sciences. 2012; 7(132):146–153. (In Russ.)
3. **Gridin V.N., Anisimov V.I., Ahmad A.D.** Building .NET client applications in distributed circuit computer-aided design. Sistemy i Sredstva Informatiki. 2016; 26(1):76–85. DOI:10.14357/08696527160106. (In Russ.)
4. **Gridin V.N., Dmitrevich G.D., Anisimov D.A.** Construction of systems of the automated designing on a basis web-technologies. Information Technologies. 2011; (5):23–26. (In Russ.)
5. **Laristov D.A., Anisimov D.A.** Development of embedded web interface for design automation systems. Izvestiya SPbGETU "LETI". Seriya: Informatika, Upravlenie i Komp'yuternye Tekhnologii. 2007; (2):63–66. (In Russ.)
6. **Anisimov D.A.** Methods of construction of systems of automation circuitry designing on the basis of web services. Izvestiya SPbGETU "LETI". 2012; (10):56–61. (In Russ.)
7. **Pissanetzky S.** Sparse matrix technology. N.Y.: Acad. Press; 1984: 321. ISBN:0-12-557580-7.

8. **Esterbyu O., Zlatev Z.** Pryamye metody dlya razrezhennykh matrits [Direct methods for sparse matrices]. Moscow: Mir; 1987: 118. (In Russ.)
9. **Norenkov I.P.** Vvedenie v avtomatizirovannoe proektirovanie tekhnicheskikh ustroystv i system [Introduction to computer aided design of technical devices and systems]. Moscow: Vysshaya Shkola; 1986: 302. (In Russ.)
10. **Anisimov V.I., Gridin V.N., Abuhazim M.M.** Data compression in automation systems based on the schematic design fixed format methods. Sistemy Vysokoy Dostupnosti. 2016; (4):34–40. (In Russ.)
11. **Knuth D.E.** The art of computer programming, Vol. 1: Fundamental algorithms. Boston: Addison-Wesley Edu. Publ.; 1968: 634. ISBN:0-201-03801-3.
12. **Vlakh I., Singkhal K.** Mashinnye metody analiza i proektirovaniya elektronnykh skhem [Computer methods for electronic circuit analysis and design]. Moscow: Radio i Svyaz'; 1988: 560. (In Russ.)
13. **Anisimov V.I., Amakhvr Yu.M.** Kompaktnye metody obrabotki razrezhennykh matrits zadach monitoringa na osnove spisochnykh struktur [Compact methods for processing sparse matrices of monitoring problems based on list structures]. Trudy Pyatoy Mezhdunarodnoy Konferentsii "Priborostroenie v Ekologii i Bezopasnosti Cheloveka". SPb.: GUAP; 2007: 157–160. (In Russ.)
14. **Amakhvr Yu.M.** Compact list-based sparse matrix processing. Herald of Computer and Information Technologies. 2008; (4):49–52. (In Russ.)
15. **Deitel Kh.M., Deitel P.Dzh.** Kak programmirovat' na Java [Java: How to program]. Moscow: Binom Press; 2003: 847. (In Russ.)
16. **Troelsen A.** Pro C# 2005 and the .NET 2.0 Platform. 3rd ed. Apress; 2005: 1032. ISBN:978-1-59059-419-3.
17. **Schildt H.** C#: The complete reference. 2nd ed. Osborne: McGraw Hill; 2002: 890. ISBN:0-07-226209-5.
18. **MacDonald M., Szpuszta M.** Pro ASP.NET 2.0 in C# 2005. 2nd ed. Apress; 2005: 1288. ISBN:1-5905-9496-7.