УДК 004.021

О. В. МИНЬКО, К. Е. ЗОЛОТЬКО

РАЗРАБОТКА ПРОГРАММНО-МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ РАЗРЕЖЕННЫХ МАТРИЦ С ПОМОЩЬЮ ТЕХНОЛОГИ OPENMP

Работа посвящена параллельному вычислению разреженных матриц. В ходе работы были спроектированы и оптимизированы с помощью языка программирования С алгоритмы для работы с разреженными матрицами для следующих операций: транспонирование, умножение разреженных матриц, умножение разреженной матрицы на плотный вектор, сложение разреженных матриц. Для хранения данных разработан гибкий алгоритм, который применяет наиболее оптимальный вариант хранения для конкретного типа операций. Были применены процедуры выделения плотных подматриц в разреженных матрицах. Для ускорения работы программы используется технология параллельного вычисления OpenMP, которая дает возможность на аппаратном уровне минимизировать ресурсы и время выполнения.

Ключевые слова: разреженные матрицы, формат crs, операции над разреженными матрицами, параллельная реализация, openmp.

Робота присвячена паралельному обчисленню розріджених матриць. В ході роботи були спроектовані і оптимізовані за допомогою мови програмування С алгоритми для роботи з розрідженими матрицями для наступних операцій: транспонування, множення розріджених матриць, множення розрідженій матриці на щільний вектор, додавання розріджених матриць. Для зберігання даних розроблений гнучкий алгоритм, який застосовує найбільш оптимальної варіант збереження для конкретного типу операцій. Були застосовані процедури виділення щільних підматриць в розріджених матрицях. Для прискорення роботи програми використовується технологія паралельного обчислення OpenMP, яка дає можливість на апаратному рівні мінімізувати ресурси і час виконання.

Ключові слова: розріджені матриці, формат crs, операції над розрідженими матрицями, паралельна реалізація, openmp.

The work is dedicated to parallel computing sparse matrices. The work has been designed and optimized using the C programming language algorithms for working with sparse matrices for the following operations: transposition, multiplication of sparse matrices and sparse matrix multiplication on a tight vector addition of sparse matrices. For data storage it developed a flexible algorithm that uses the most appropriate option to save for a specific type of operation. Selection procedures were applied in dense submatrices sparse matrices. To speed up the program using parallel computing technology OpenMP, which makes it possible to minimize the hardware resources and execution time.

Keywords: sparse matrices, crs format operations on sparse matrices, parallel implementation, openmp.

Введение. Будем рассматривать произвольные разреженные матрицы (sparse matrix). Разреженная матрица определяется как матрица с преимущественно нулевыми элементами. Среди специалистов нет единства в определении того, какое именно количество ненулевых элементов делает матрицу разреженной. Одним из возможных вариантов для m×n матриц является o(m*n) ненулевых элементов.

Методы хранения и обработки разреженных матриц в течение последних десятилетий вызывают интерес у широкого круга исследователей [1–4]. В ходе работы было разработано программно-математическое обеспечение для работы с основными операциями с разреженными матрицами и применены гибкие процедуры выделения плотных матриц с целью повышения производительности разработанного программного обеспечения.

В качестве программного средства для распараллеливания и ускорения программного обеспечения используется технология OpenMP.

Цель и задачи исследования. Качестве исходной конфигурации будем рассматривать произвольную разреженную матрицу $A^{m\times n} = ||A_{ij}||$ (вообще говоря m не равно n) элементы которой представляют собой элементы множества {0,1}. Если интерпретировать матрицу как матрицу «субъекты-объекты», то множество U (множество строк матрицы) будет представлять собой множество субъектов R множество - множество объектов.

Как правило, ненулевые элементы не образуют чёткой структуры и представляют собой случайно разбросанные точки. Однако, возможно существуют такие перестановки строк $\{u_{i1}, u_{i2}, \dots, u_{im}\}$ и столбцов $\{r_{j1}, r_{j2}, \dots, r_{jn}\}$ исходной матрицы, что подмножества единичных элементов будут образовывать сгущения некоторой формы. Назовём эквивалентными преобразованиями матрицы подстановки местами её строк или столбцов.

Цель данной работы состоит в разработке и анализе алгоритмов параллельной обработке разреженных матриц, а также исследование методов, которые эквивалентными преобразованиями перемещают единичные элементы так, чтобы сформировать некоторые конфигурации из единичных элементов, которые будут образовывать «плотные» подматрицы.

Формат хранения данных. Для хранения разреженной матрицы в работе будет применяться широко распространенный формат под названием CSR (Compressed Sparse Rows) или CRS (Compressed Row Storage), который использует три массива. Первый массив хранит значения элементов построчно (строки рассматриваются по порядку сверху вниз), второй – номера столбцов для каждого элемента, а третий – индексы первых элементов каждой строки.

Оценим объем памяти, необходимый для хранения разреженной матрицы в формате CRS: M = 8 NZ + 4 NZ + 4 (N + 1) = 12 NZ + 4 N + 4.

Пример изображен на рис. 1.

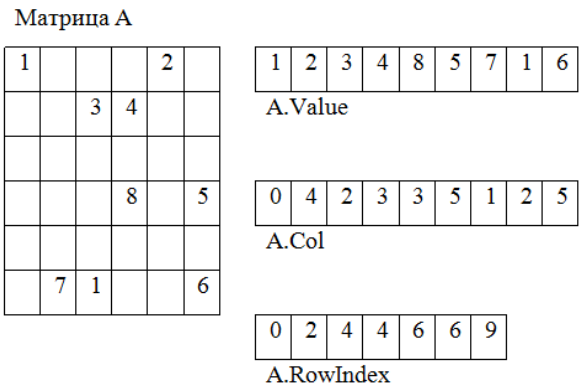

Рис. 1 – Формат CRS

Генерация модельных данных. Для проведения экспериментов был написан генератор модельных данных, работа которого заключается в следующем: генерируются случайные разреженные матрицы со сгущениями, размеры которых *m* и *n* лежат в некотором диапазоне.

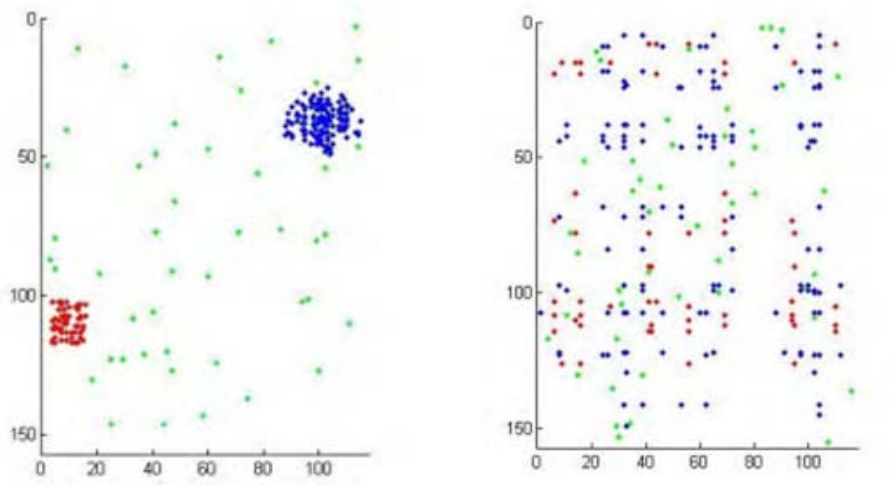

Рис. 2 – Пример сгенерированной разреженной матрицы с двумя сгущениями после третьего и четвёртого этапов (матрицы A и B)

Генерация осуществляется следующей последовательностью действий:

- сгенерировать матрицу с равномерно распределёнными ненулевыми элементами;
- выбрать несколько случайных подматриц размеры, которых лежат в некотором диапазоне; при этом порядок размеров подматриц меньше порядка матрицы, например $O\left(\sqrt{n}\right)$ или $O\left(n^{\alpha}\right)$, где $\alpha \le 1-\varepsilon$, где $\varepsilon \ge 0$ - некоторый порог;
- заполнить данные подматрицы некоторым количеством единиц; заполнение можно вести по определённому закону с определённой степенью интенсивности заполнения (например, можно использовать алгоритмы дискретизации изображений (Брезенхэма и др.), [7] (обозначим: A - матрица, полученная на данном этапе);
- осуществить большое случайных число ранспозиций строк и столбцов (обозначим: B - матрица, полученная на данном этапе).

Пример генерации матрицы представлен на рис. 2. Элементы, принадлежащие разным сгущениям, визуализированы разными цветами. Шумовые элементы визуализированы зелёным цветом. Размер матрицы 153×112, число ненулевых элементов – 233 .

Алгоритм выделения плотных матриц. Идея данного алгоритма выделения сгущений заключается в том, чтобы собрать ненулевые элементы как можно ближе к главной диагонали. Введём матрицу весов элементов $W^{m \times n}$. Зададим значение элементов таким образом, чтобы они возрастали по мере удаления от главной диагонали (например, линейно, $a_i = k|i|$, или экспоненциально, $a_i = a^{|i|}$, причём положим $a_0 = 0$), матрица B. Умножим поэлементно матрицу B на матрицу W, просуммируем все элементы текущей матрицы, получим вес текущей конфигурации. Также введём переменную-счётчик k. На каждом шаге будем генерировать случайную перестановку из двух номеров строк или столбцов.

Очевидно, что вес будет уменьшаться в том случае, если единичные элементы будут располагаться ближе к центральной диагонали матрицы. Возможны некоторые модификации данного алгоритма, основанные на других методах глобальной оптимизации. На рис. 3 приведены результаты работы данного алгоритма (визуализированы только исходная и конечная матрицы). Как можно заметить из экспериментов, сгущения получаются более разреженные, чем при работе алгоритмов, основанных на сходстве.

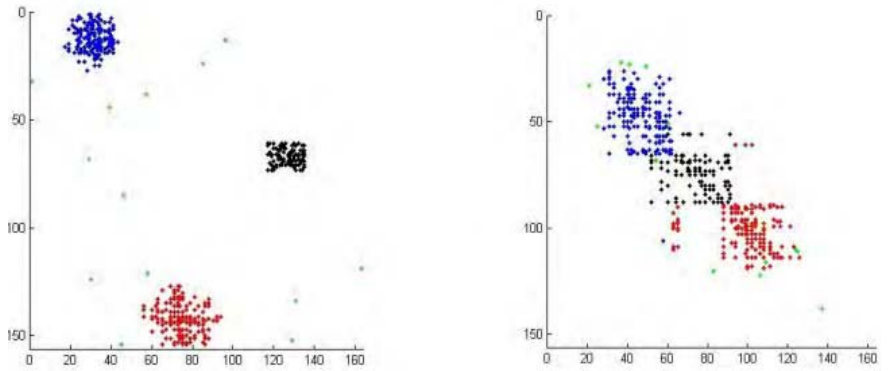

Рис. 3 – Результат выделения плотных матриц

Умножения разреженной матрицы на разреженную матрицу и параллельная реализация. Пусть дано матрица A_{pxq} и матрица B_{qxr}. Результат умножения матрица C_{pxr} имеет элементы изображены на (1).

$$c_{ik} = \sum_{j=1}^{q} a_{ij} b_{jk},\ i = 1, \ldots, p.k = 1, \ldots,\ r. \qquad (1)$$

Эта формула выражает элемент c_{ik} как скалярное произведение i-й строки A и k-го столбца B. Однако, если B задана строчным форматом, то к ее столбцами нет прямого доступа. Одно возможное решение этой проблемы заключается в транспонировании B. Тогда уравнение (1) будет иметь вид (2).

$$c_{ik} = \sum_{j=1}^{q} a_{ij} \left(B^{T}\right)_{kj}, \qquad (2)$$

и использует только строки A и B^T.

Учитывая, что C хранится в формате CRS, важно избежать переупаковок. Для этого нужно обеспечить пополнение матрицы C ненулевыми элементами последовательно, по строкам - слева направо, сверху вниз. Нужно брать первую строчку матрицы A и умножать ее поочередно на все столбцы матрицы B (строки матрицы B^T). В этом случае обеспечивается последовательное пополнение матрицы C, что позволяет дописывать элементы в массивы Values и Cols, а также формировать массив RowIndex.

Алгоритм умножения разреженной матрицы на разреженную матрицу:

1. Создать 2 вектора (Value, Col) и массив RowIndex длины N + 1 для сохранения матрицы C.
2. транспонировать матрицу B.
3. в цикле по i от 0 до N - 1 перебирайте все строки матрицы A. Для каждого i в цикле по j от 0 до N - 1 перебираем все строки матрицы BT. Вычислить скалярное произведение векторов - строк Ai и Bj, пусть оно равно V. Если V отлично от нуля, то добавляем в вектор Value элемент V, а в вектор Col - элемент j. При окончании цикла по j, копируем значение RowIndex [i + 1], записав туда текущее значение числа ненулевых элементов V.

Результаты исследования. Проведя эксперименты, используя для вычисления матрицы разного порядка и сравнивая время выполнения работы программы на 1 потоке, на 8 потоках и с помощью библиотеки MKL. Результаты эксперимента приведены на рис. 1. и рис. 2.

Проанализировав полученные результаты приходим к выводу, что наша реализация работает медленнее для матриц, порядок которых меньше 20000, а переходя этот порядок идет выигрыш во времени для 8 потоков.

Вычислительные эксперименты проводились с использованием следующих инфраструктур, характеристики которых приведены в табл. 1. и табл. 2

Таблица 1 – Тестовая инфраструктура №1

Процессор	Intel(K) core(TM) i7-3632QM CPU @ 2.20Ghz
Память	8 Gb
Программная среда	Visual Studio Premium 2012
ОС	Windows 8 x64

Результаты работы программы на тестовой инфраструктуре № 1

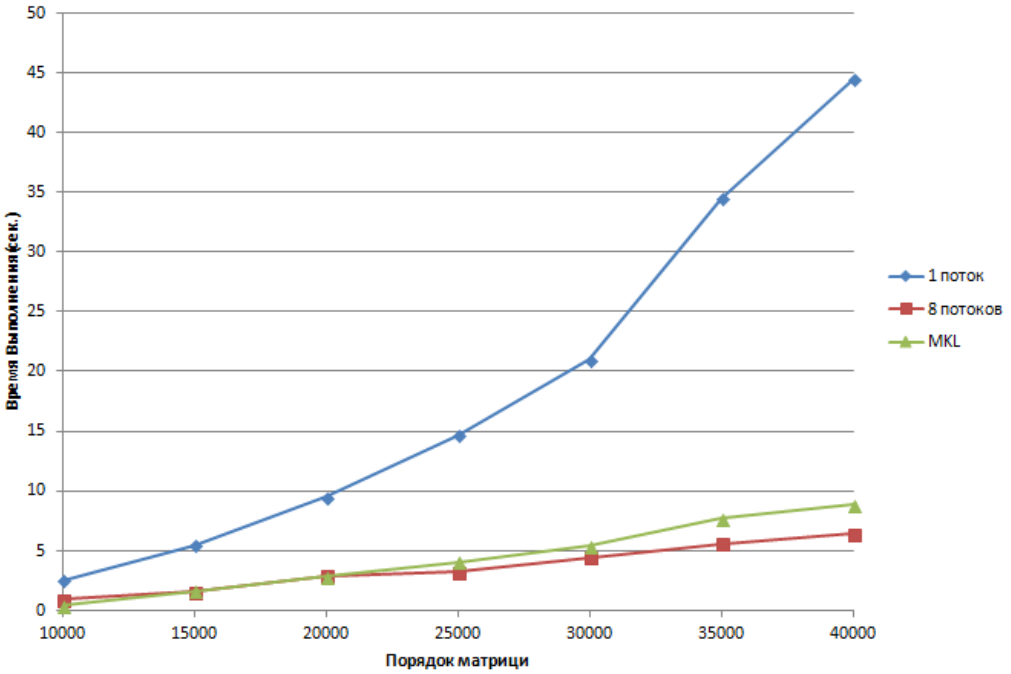

Рис. 4 – Сравнение времени умножения разреженных матриц (инфраструктура №1)

Проанализировав полученные результаты приходим к выводу, что наша реализация работает медленнее для матриц, порядок которых меньше 20000, а переходя этот порядок идет выигрыш во

времени для 8 потоков,за счет оптимизации и минимизации внутренних операций обработки матриц. Подробнее приведенную зависимость, которую можно увидеть на рис. 4.

Проведем вычислительные эксперименты для описанной задачи и сравним полученные результаты на более слабой тестовой инфраструктуре. И проследим в какой степени зависят реализованные методы от инфраструктуры и производительности системы для получение более объективной картины исследований.

Таблица 2 – Тестовая инфраструктура №2

Процессор	Intel® Core™ i5-5250U Processor (3M Cache, up to 2.70 GHz)
Память	4 Gb
Программная среда	Visual Studio Premium 2012
ОС	Windows 8.1 x64

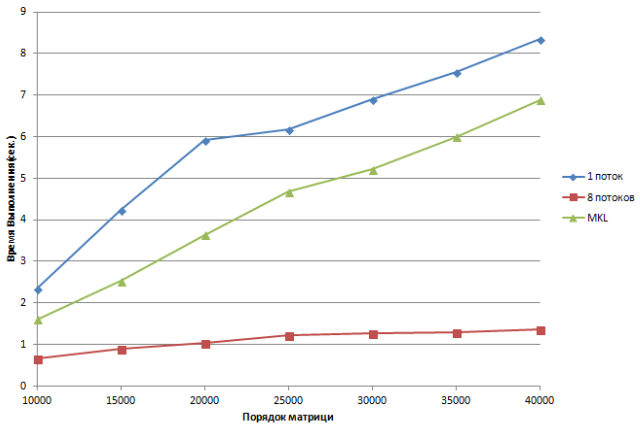

Рис. 5 – Сравнение времени умножения разреженных матриц (инфраструктура №1)

Аналогично проанализировав полученные результаты приходим к выводу, что наша реализация работает медленнее для 8 потоков любого порядка матриц чем реализация в MKL.

На рис. 5. показано решении задачи с помощью разработанных реализаций с использованием разного количества потоков, реализации MKL и последовательной реализации.

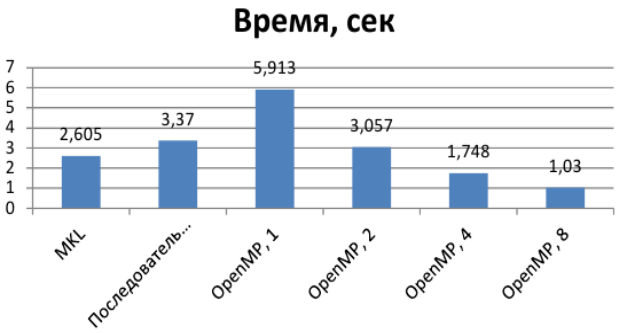

Рис. 6 – Сравнение времени умножения разреженных матриц на основе различного количества потоков реализации MKL и последовательной реализации.

На рис. 6 показано время решения задачи с помощью разработанных реализаций. Во-первых, отметим тот факт, что однопоточная OpenMP-реализация (OpenMP, 1) работает в 1.75 раза медленнее последовательной.

Предположительно такой эффект связан с тем, что компилятор для последовательной версии формирует более оптимальный код. Проверка или опровержение данной гипотезы требует дополнительных исследований, что выходит за рамки лабораторной работы. Во-вторых, OpenMP-реализация имеет не совсем линейную масштабируемость. Параллельная реализация, запущенная в 2 потока (OpenMP, 2), работает примерно в 1.93 раза быстрее однопоточной версии, в 4 потока (OpenMP, 4) – 3.38 раза, в 8 потоков (OpenMP, 8) – 5.74. Одна из возможных причин плохой масштабируемости – это неэффективная работа с памятью, т.к. элементы строки распределены по всей длине, поэтому потоки обращаются к ячейкам массива в произвольном порядке.

Результаты работы алгоритма выделения плотных подматриц.

Протестируйте алгоритм с разными порогами сходства на матрице размера 9424 × 31867,число ненулевых элементов 11549. результаты приведены в табл. 3.

Как можно видеть, с ростом порога сходства плотность подматрицы максимального размера увеличивается, однако уменьшается её размер рис. 7.

Таблица 3 – Результаты работы алгоритма выделения плотных подматриц

Пороги сходства по строкам и столбцам	Число строк	Чмсло столбцов	Плотность
(2,2)	313	481	0.0158
(2,3)	132	90	0.0573
(2,4)	36	24	0.2174
(2,5)	18	6	0.4259
(2,6)	13	3	0.7179
(3,2)	178	481	0.0242
(3,3)	62	90	0.0935
(3,4)	24	24	0.2413
(3,5)	6	6	0.5833
(3,6)	13	3	0.7179

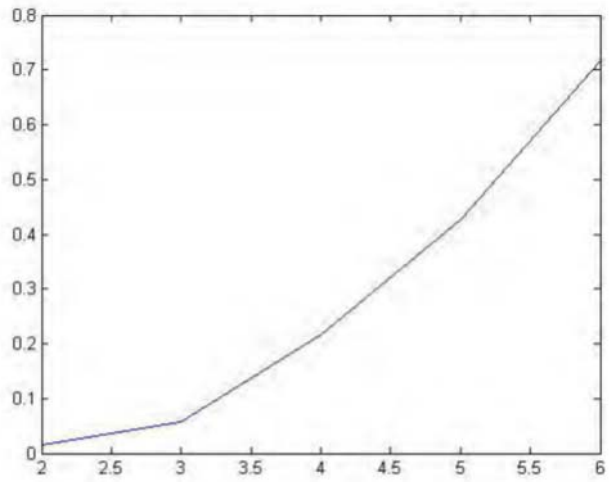

Рис. 7 – Зависимости размеров сгущения с увеличением порога

Выводы. При анализе результатов было замечено, что специальные алгоритмы обработки разреженных матриц работают гораздо быстрее их аналоги для обработки обычных (плотных матриц) рис. 1. и рис. 2. чем больше разреженность входных матриц, тем эффективнее работают реализованы алгоритмы. И наоборот, чем более плотная матрица, тем эффективнее будут стандартные алгоритмы ее обработки. В общем случае следует учитывать структуру матрицы перед тем, как использовать тот или иной подход для ее обработки.

Полученные результаты могут быть использованы при решении задач линейной и нелинейной оптимизации, где входными параметрами являются разреженные матрицы. Еще одной из сфер применения является многочисленные реализации методов математической физики, а именно при применении: метода конечных элементов; метода сеток.

Список использованной литературы:

1. *Писсанецкы, С.* Технология разреженных матриц [Текст] / *С. Писсанецкы.* − Москва: Мир, 1988. − 410 с.
2. *Джордж, А.* Численное решение больших разреженных систем уравнений [Текст] / *А. Джордж, Дж. Лю*. − Москва: Мир, 1984. − 333 с.
3. *Тьюарсон, Р.* Разреженные матрицы [Текст] / *Р. Тьюарсон*. − Москва: Мир, 1977. – 191 с.
4. *Хортон, А.* Visual C++ 2010: полный курс [Текст] / *А. Хортон*. – Диалектика, 2011. – 1216 с.
5. *Антонов, А. С.* Паралельне программирования с использованием технологии OpenMP [Текст]: учебное пособие / *А. С. Антонов.* − Москва: МГУ, 2009. – 77 с.
6. *Голуб, Дж.* Матричные вычисления [Текст] / *Дж. Голуб, Ч. Ван Лоун*. − Москва: Мир, 1999. − 548 с.
7. *Потемкин, В. Г.* Справочник по MATLAB [Электронный ресурс] / *В. Г. Потемкин.* – 164 с. – Режим доступа: \www/URL: http://ui-engineers.ddns.net/_ld/1/143___Matlab.pdf
8. *Касперски, К.* Техника оптимизации программ. Эффективное использование памяти [Текст] / *К. Касперски.* − Санкт-Петербург: БХВ-Петербург, 2003. – 464 с.
9. *Хортон, А.* Visual C++ 2005: базовый курс [Текст] / *А. Хортон*. – Вильямс, 2007. – 1152 с.
10. *Demmel, J. W.* Applied Numerical Linear Algebra [Text] / *J. W. Demmel.* − SIAM, 1997. − 431 p.
11. *Demmel, J. U. C.* Applications of Parallel Computers [Electronic resource] / *J. U. C Demmel.* – Spring, 2014. − Available at: \www/URL: https://people.eecs.berkeley.edu/~demmel/cs267_Spr14/
12. *Tewarson, R. P.* Sparse Matrices [Text] / *R. P. Tewarson*. − Academic Press, 1973. – 160 p.
13. *Buttari, A.* A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures [Text]: technical report / *A. Buttari, J., Langou, J., Kurzak, J. Dongarra*. − University of Tennessee, 2007. – 191 p.
14. *Керниган, Б.* Язык программирования С [Текст] / *Б. Керниган, Д. Ритчи.* − Вильямс, 2009. – 304 с.
15. *Гербер, Р.* Оптимизация ПО. Сборник рецептов [Текст] / *Р. Гербер, А. Бик, К. Смит, К. Тиан*. – Санкт Петербург: Питер, 2010. – 352 с.
16. *Bik, A. J. C.* The Software Vectorization Handbook: Applying Multimedia Extensions for Maximum Performance [Text] / *A. J. C. Bik.* − Intel Press, 2006. – 236 p.
17. *Воронцов, К. В.* Математические методы обучения по прецедентам (теория обучения машин) [Электронный ресурс]: курс лекций / *К. В. Воронцов*. – 141 с. – Режим доступа: \www/URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf
18. *Дьяконов, А. Г.* Анализ данных, обучение по прецедентам, логические игры, системы WEKA, RapidMiner и MatLab [Текст] / *А. Г. Дьяконов.* – МАКСПресс, 2010. – 278 с.
19. *Журавлёв, Ю. И.* Избранные научные труды [Текст] / *Ю. И. Журавлёв.* − Москва: Магистр, 1998. – 420 с.
20. *Журавлёв, Ю. И.* Корректные алгебры над множествами некорректных (эвристических) алгоритмов [Текст] / *Ю. И. Журавлёв* // Кибернетика. − 1978. − № 2. − С. 35–43.
21. *Рудаков, К. В.* Об алгебраической теории универсальных и локальных ограничений для задач классификации [Текст] / *К. В. Рудаков* // Распознавание, классификация, прогноз. – 1989. – № 1. – С. 176−201.
22. *Рудаков, К. В.* Полнота и универсальные ограничения в проблеме коррекции эвристичских алгоритмов классификации [Текст] / *К. В. Рудаков* // Кибернетика. − 1987. − № 3. − С. 106−109.
23. *Петров, С. О.* Дослідження застосування операторів згортки в задачах виділення границь на зображенні [Текст] / *С. О. Петров, І. О. Марченко, Б. О. Дібров* // Східно-Європейський журнал передових технологій. – 2015. − № 6/4 (78). – С. 27−31. doi:10.15587/1729-4061.2015.56548

Bibliography (transliterated):

1. Pissanetski, S. (1988). Sparse matrix technology. Moscow: Mir, 410.
2. George, A., Liu, J. (1984). Numerical solution of large sparse systems of equations. Moscow: Mir, 333.
3. Tewarson, R. (1977). Sparse matrix. Moscow: Mir, 191.
4. Horton, I. (2011). Visual C++ 2010: full course. Dialektika, 1216.
5. Antonov, A. S. (2009). Paralelne programming using technology OpenMP. Moscow: MGY, 77.
6. Golub, J., Van Loan, C. (1999). Matrix calculations. Moscow: Mir, 548.
7. Potemkin, V. G. Guide to the MATLAB, 164. Available at: http://ui-engineers.ddns.net/_ld/1/143___Matlab.pdf
8. Kaspersky, K. (2003). Technique optimization programs. Efficient use of memory. Saint Petersburg: BHV-Petersburg, 464.
9. Horton, I. (2007). Visual C++ 2005: full course. Williams, 1152.
10. Demmel, J. W. (1997). Applied Numerical Linear Algebra. SIAM, 431.
11. Demmel J. U.C. (2014). Applications of Parallel Computers. Available at: https://people.eecs.berkeley.edu/~demmel/cs267_Spr14/
12. Tewarson, R. P. (1973). Sparse Matrices. Academic Press, 160
13. Buttari, A., Langou, J., Kurzak, J., Dongarra, J. (2007). A Classof Parallel Tiled Linear Algebra Algorithms for Multicore Architectures. University of Tennessee, 191.
14. Kernighan, B., Ritchie, D. (2009). The C Programming Language. Williams, 304.
15. Gerber, R., Bik, A. J. C., Smith, K. B., Tian, X. (2010). The Software Optimization Cookbook. Saint Petersburg: Piter, 352
16. Bik, A. J. C. (2006). The Software Vectorization Handbook: Applying Multimedia Extensions for Maximum Performance. Intel Press, 236.
17. Vorontsov, K. V. Mathematical methods of training on precedents (machine learning theory), 141. Available at: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf
18. Deacons, A. (2010). The data analysis, training on precedents, logic games, WEKA system, RapidMiner and MatLab. MAKSPress, 278.
19. Zhuravlev, Y. (1998). Selected scientific works. Moscow: Magistr, 420.
20. Zhuravlev, Y. (1978). Correct algebras over sets incorrect (heuristic) algorithms. Cybernetics, 2, 35−43.
21. Rudakov, K. (1989). On the algebraic theory of universal and local constraints for classification problems. Recognition, classification, prognosis, 1, 176−201.

22. Rudakov, K. (1987). Completeness and universal constraints in the problem of correction heuristic algorithms cal classification. Cybernetics, 3, 106−109.

23. Petrov, S. O., Marchenko, I. O., Dibrov, B. O. (2015). The use of convolution operators in the tasks of edge detection. Eastern-European Journal of Enterprise Technologies, 6(4(78)), 27−31. doi:10.15587/1729-4061.2015.56548

Поступила (received) 08.01.2016

Бібліографічні описи / Библиографические описания / Bibliographic descriptions

Розробка програмно-математичного забезпечення для паралельної обробки розріджених матриць за допомогою технологи OpenMP/ О. В. Мінько, К. Є. Золотько// Вісник НТУ «ХПІ». Серія: Механіко-технологічні системи та комплекси. – Харків : НТУ «ХПІ», 2016. – No 4(1176). – С.44–49. – Бібліогр.: 23 назв. – ISSN 2079-5459.

Разработка программно-математического обеспечения для параллельной обработки разреженных матриц с помощью технологи OpenMP/ О. В. Минько, К. Е. Золотько// Вісник НТУ «ХПІ». Серія: Механіко-технологічні системи та комплекси. – Харків : НТУ «ХПІ», 2016. – No 4(1176). – С.44–49. – Бібліогр.: 23 назв. – ISSN 2079-5459.

The development of mathematical software for parallel processing of sparse matrices using OpenMP technology/ O. V. Minko, K. E. Zolotko//Bulletin of NTU "KhPI". Series: Mechanical-technological systems and complexes. – Kharkov: NTU "KhPI", 2016. – No 4 (1176) .– P.44 –49. – Bibliogr.: 23. – ISSN 2079-5459.

Відомості про авторів / Сведения об авторах / About the Authors

Минько Олег Владимирович – магистр, кафедра компьютерных технологий, Днепропетровский национальный университет имени Олеся Гончара, пр. Гагарина, 72, г. Днепр, Украина, 49010; e-mail: minko.oleg@gmail.com.

Золотько Константин Евгеньевич – кандидат технических наук, доцент, кафедра компьютерных технологий, Днепропетровский национальный университет имени Олеся Гончара, пр. Гагарина, 72, г. Днепр, Украина, 49010; e-mail: zolt66@gmail.com.

Мінько Олег Володимирович – магістр, кафедра комп'ютерних технологій, Дніпропетровський національний університет імені Олеся Гончара, пр. Гагаріна, 72, м Дніпро, Україна, 49010; e-mail: minko.oleg@gmail.com.

Золотько Костянтин Євгенійович – кандидат технічних наук, доцент, кафедра комп'ютерних технологій, Дніпропетровський національний університет імені Олеся Гончара, пр. Гагаріна, 72, м Дніпро, Україна, 49010; e-mail: zolt66@gmail.com.

Minko Oleh Volodymyrovych – master, department of computer technology, Oles Honchar Dnipropetrovsk National University, Gagarina, 72, of the Dnipro, Ukraine, 49010;. e-mail: minko.oleg@gmail.com.

Zolotko Konstantin Evgenevich – Ph.D., Associate Professor, Department of Computer Technologies, Oles Honchar Dnipropetrovsk National University, Gagarina, 72, of the Dnipro, Ukraine, 49010;. e-mail: zolt66@gmail.com.

УДК 656.212:681.3

Н. В. МОСКАЛЕЦ

МЕТОДЫ ОРГАНИЗАЦИИ ПРОСТРАНСТВЕННО-ВРЕМЕННОГО ДОСТУПА В СИСТЕМЕ МОБИЛЬНОЙ СВЯЗИ

Розглядаються питання організації просторово-часового доступу в системах мобільного зв'язку на основі адаптивної антеною решітки і алгоритмів просторово-часової обробки з оптимальною процедурою розрахунку вектора вагових коефіцієнтів. Проводиться оцінка ефективності адаптивної антенної решітки за обраним критерієм мінімального середньоквадратичного відхилення. Отримано результати коефіцієнта ступеня зменшення сумарних завад на виході адаптивної антеною решітки з оптимальними ваговими коефіцієнтами в сталому режимі та незмінній сигнально-завадовій обстановки.

Ключові слова: просторово-часовий доступ, адаптивна антенна решітка, вагові коефіцієнти, просторово-часова обробка.

Рассматриваются вопросы организации пространственно-временного доступа в системах мобильной связи на основе адаптивной антенной решетки и алгоритмов пространственно-временной обработки с оптимальной процедурой расчета вектора весовых коэффициентов. Проводится оценка эффективности адаптивной антенной решетки по выбранному критерию минимального среднего квадратического отклонения. Получены результаты коэффициента степени подавления суммарных помех на выходе адаптивной антенной решетки с оптимальными весовыми коэффициентами в установившемся режиме и неизменной сигнально-помеховой обстановке.

Ключевые слова: пространственно-временной доступ, антенная решетка, весовые коэффициенты, пространственно-временная обработка.

It is considered the analysis of methods organization space-time multiple access (SDMA) are in mobyle communication.
Ability is shown on the Implementation of the given method is based on N-elements adaptive antenna array independently from second used purposing methods, something result attraction extended resource spatial-time parameters.
We studied a method of organizing SDMA for each subscriber station for access to the resources of the base station in a mobile communication system using the group receiving adaptive antenna array (AAA).
The proposed method consists in the formation of the individualdistribution structure of the field received on the basis of the optimum evaluation procedure signal weight vector by the criterion of the minimum mean square deviation.