УДК 004.023
DOI: 10.25559/SITITO.15.201903.619-625

Анализ алгоритма оптимального распределения

В. И. Мунерман*, Д. В. Мунерман
Смоленский государственный университет, г. Смоленск, Россия
214000, Россия, г. Смоленск, ул. Пржевальского, д. 4
* vimoon@gmail.com

Аннотация

В статье рассматривается один алгоритм оптимального распределения «объектов» произвольной природы по «хранилищам», сущность которых определяется предметной областью. Рассматриваются некоторые предметные области, для которых актуальна проблема оптимального распределения. Ускорение операции Join для больших данных за счет параллельной реализации операции Join необходимо равномерное распределение данных между процессорами кластера. В этом случае параллельная реализация операции Join будет эффективной только тогда, когда вычислительные сложности ее выполнения во всех фрагментах базы данных будут минимально отличаться друг от друга. Критерий оптимальности должен обеспечить равномерность распределения данных. В складской логистике рассматривается не стандартная задача, состоящая в минимизации количества хранилищ или максимизации «веса» (числа, массы, площади или объема) загруженных объектов (товаров, материалов). Решаемая задача состоит в размещении объектов по хранилищам таким образом, чтобы сумма «свободных мест» в хранилищах была минимальной. Задача образовательной логистики заключается в «справедливом» распределении бюджетных мест с учетом направления подготовки, требований регионов и возможностей университетов. Она характеризуется большим набором ограничений, которые определяют минимальное и максимальное количество требуемых бюджетных мест для каждого региона и университета. Приведено подробное описание эвристического алгоритма оптимального распределения. Предложены целевые функции для рассматриваемых задач. Приведено описание экспериментов, которые позволили дать оценку качества эвристического жадного алгоритма оптимального распределения. В результате этих экспериментов были выявлены зависимости времени выполнения алгоритма от количества распределяемых объектов и качества распределения (разности между максимумом и минимумом заполнения хранилищ) от количества хранилищ и интервала значений весов объектов. Показано, что алгоритм достаточно прост и может быть легко реализован в любом языке программирования. Время работы алгоритма даже для big data мало, что позволяет эффективно использовать его при подготовке данных для параллельного решения задач с высокой вычислительной сложностью. алгоритм показывает хорошие результаты при распределении объектов по хранилищам. Наибольшее заполнение хранилищ отличается от наименьшего на незначительную величину.

Ключевые слова: оптимальное распределение, параллельное программирование, эвристический алгоритм.

Финансирование: исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований в рамках научного проекта № 19-05-00231а «Пространственная организация высшей школы и региональное развитие: из прошлого в будущее».

Для цитирования: *Мунерман В. И., Мунерман Д. В.* Анализ алгоритма оптимального распределения // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 3. С. 619-625. DOI: 10.25559/SITITO.15.201903.619-625

Analysis of an Optimal Distribution Algorithm

V. I. Munerman[*], **D. V. Munerman**
Smolensk State University, Smolensk, Russia
4 Przhevalsky Str., Smolensk 4214000, Russia
[*] vimoon@gmail.com

Abstract

The article considers one algorithm for the optimal distribution of "objects" of an arbitrary nature among "storages", the essence of which is determined by the subject area. Some subject areas for which the optimal distribution problem is relevant are considered. Accelerating the Join operation for big data due to the parallel implementation of the Join operation requires uniform distribution of data between the cluster processors. In this case, parallel implementation of the Join operation will be effective only when the computational complexity of its execution in all database fragments will be minimally different from each other. The optimality criterion should ensure uniform distribution of data. In warehouse logistics, a non-standard task is considered, consisting in minimizing the number of storages or maximizing the "weight" (number, mass, area or volume) of loaded objects (goods, materials). The task at hand is to place objects in storage in such a way that the amount of "free space" in the storage is minimal. The task of educational logistics is to "equitably" allocate budget places, taking into account the direction of training, the requirements of the regions and the capabilities of universities. It is characterized by a large set of restrictions that determine the minimum and maximum number of required budget places for each region and university. A detailed description of the heuristic optimal distribution algorithm is given. Objective functions for the problems under consideration are proposed. A description is given of the experiments that made it possible to assess the quality of the heuristic greedy optimal distribution algorithm. As a result of these experiments, the dependences of the execution time of the algorithm on the number of distributed objects and the quality of distribution (the difference between the maximum and minimum storage capacity) on the number of stores and the interval of the values of the object weights were revealed. It is shown that the algorithm is quite simple and can be easily implemented in any programming language. The running time of the algorithm, even for big data, is small, which allows it to be effectively used in the preparation of data for parallel solving problems with high computational complexity. The algorithm shows good results when distributing objects across repositories. The largest storage capacity differs from the smallest by a small amount.

Keywords: optimal distribution, parallel programming, heuristic algorithm

Funding: The study was supported by the Russian Foundation for Basic Research as a part of the scientific project No. 19-05-00231a "Spatial Organization of Higher Education and Regional Development: from the Past to the Future".

For citation: Munerman V.I., Munerman D.V. Analysis of an Optimal Distribution Algorithm. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2019; 15(3):619-625. DOI: 10.25559/SITITO.15.201903.619-625

1. Проблема оптимального распределения

В статье рассматривается один алгоритм оптимального распределения «объектов» произвольной природы по «хранилищам», сущность которых определяется предметной областью. Далее рассматриваются некоторые предметные области, для которых актуальна проблема оптимального распределения.

1.1. ***Ускорение операции Join для больших данных***

В этом случае для параллельной реализации операции Join необходимо равномерное распределение данных между процессорами кластера. Это позволяет уравнять скорости обработки каждым из них своего фрагмента данных. Две таблицы, операнды этой операции, распределяются в соответствии с принципом, который получил название «симметричное горизонтальное распределение» (СГР) [1, 2]. В этом случае таблицы представляют собой фактормножества, состоящие из классов эквивалентности, каждый из которых содержит строки таблиц с одним и тем же значением ключа, заданного в опции ON. Для получения результата операции Join необходимо выполнить декартовы произведения всех пар классов эквивалентности обеих таблиц, которые соответствуют друг другу. Соответствие состоит в том, что все строки классов эквивалентности пары содержат одно и то же значение ключа. При таком подходе таблицы разбиваются на фрагменты, которые расположены в фрагментах основной базы данных. Каждый фрагмент БД обрабатывается отдельным процессором кластера.

Требование СГР состоит в том, чтобы:

Каждый класс эквивалентности полностью располагался в одном фрагменте таблицы;

Соответствующие классы эквивалентности обеих таблиц располагались в одном и том же фрагменте БД.

Пусть T1 и T2 таблицы операнды операции Join, K – ключ, по которому выполняется операция, K* – фиксированное значение ключа, $T1_{K*}$, $T2_{K*}$ – классы эквивалентности, соответствующие этому значению ключа, $m(T1_{K*})$, $m(T2_{K*})$ – количество строк таблиц T1 и T2 в соответствующих классах эквивалентности. Тогда вычислительная сложность декартова произведения этих классов эквивалентности имеет порядок $O(m(T1_{K*})m(T2_{K*}))$. Общая вычислительная сложность операции Join над фрагментами таблиц фрагментами T1 и T2 равна $\sum_{i=1}^{n} O(m(T1_{K_i^*}) \times m(T2_{K_i^*}))$, ($m$ – количество классов эквивалентности в этих фрагментах).

Очевидно, что параллельная реализация операции Join будет эффективной только в том случае, когда вычислительные сложности ее выполнения во всех фрагментах базы данных будут минимально отличаться друг от друга. В этом случае критерий оптимальности должен обеспечить равномерность распределения данных. В этом случае в роли объектов выступают классы эквивалентности, а в роли хранилищ – хранилища данных процессоров.

1.2. ***Складская логистика***

В этом случае рассматривается не стандартная задача, состоящая в минимизации количества складов (хранилищ) или максимизации «веса» (числа, массы, площади или объема) загруженных объектов (товаров, материалов). Как и в пункте 1.1 предполагается, что все вместе хранилища способны вместить все размещаемые объекты, но отличие состоит в том, что вводятся ограничения на объемы отдельных хранилищ. Задача состоит в размещении объектов по хранилищам таким образом, чтобы сумма «свободных мест» в хранилищах была минимальной.

1.3. ***Образовательная логистика*** **(*распределение бюджетных мест*)**

В этом случае речь идет о «справедливом» распределении бюджетных мест (объектов) с учетом направления подготовки, требований регионов и возможностей университетов (хранилищ). Предполагается все объекты имеют одинаковые «веса». Как правило это количество бюджетных мест, кратное числу студентов в одной учебной группе. Для каждого направления подготовки выделяется определенное количество бюджетных мест, определяемое экономическими требованиями государства. Это означает, что количество бюджетных мест, выделяемых на каждое направление подготовки ограничено сверху. Для каждого региона и задается ограничения, определяющие минимальное и максимальное количество требуемых для его развития бюджетных мест, а для каждого университета максимальное количество определяется его возможностями по организации учебного процесса. Таким образом, в рассматриваемой ситуации задача равномерного распределения аналогична задаче из пункта 1.2, но с учетом существенно большего числа ограничений.

1.4. ***Анализ решения аналогичных задач***

Решение проблемы равномерного распределения объектов по хранилищам данных похоже на решение известных задач: задачи о множественном рюкзаке (0-1 Multiple Knapsack Problem) или задачи упаковки контейнеров и загрузки складов (Bin Packing and Cutting Stock Problems) [3-7]. Ее точное решение, также как точное решение указанных задач относится к классу NP. Поэтому, так как количество объектов может быть очень большим, исчисляться тысячами, и, особенно, в задаче из пункта 1.1, где количество классов эквивалентности может исчисляться миллионами и даже миллиардами, необходимо разработать достаточно быстрый алгоритм, который даст решение близкое к точному.

Следует отметить, что в области вычислительной техники и параллельных вычислений многие современные решения аналогичной проблемы эффективного использования виртуальных машин в облачных системах основаны на применении различных алгоритмов упаковки контейнеров [8, 9].

Для быстрого решения оптимального распределения для больших данных в статье предлагается эвристический алгоритм, который имеет полиномиальную вычислительную сложность. Проведена формализация задачи: задаются целевая функция и система ограничений. Приводится анализ предложенного алгоритма для различных объемов данных и количеств хранилищ данных.

2. Формальное описание задачи оптимального распределения

В этом разделе дано формальное описание задачи оптимального распределения, даны необходимые обозначения и представлен алгоритм ее решения.

Пусть нам дан набор из n элементов, каждый из которых имеет вес w_j (j = 1, ..., n), и набор из m хранилищ, каждое из которых способно вместить необходимый объем данных. Значение переменной $x_{ij} \in \{0,1\}$ x_{ij} = Кроме того должно выполняться

1 означает, что элемент j размещен в хранилище i (i=1,...,m; j=1,...,n), в противном случае x_{ij}=0. Кроме того должно выполняться условие: если $x_{ij} = x_{kj}$ то i=k, которое означает, что один объект может располагаться только в одном хранилище. Тогда целевая в общем виде функция имеет вид:

Minimize

$$z = Max(\sum_{j=1}^{n} w_j x_{1j}, ..., \sum_{j=1}^{n} w_j x_{mj}) - Min(\sum_{j=1}^{n} w_j x_{1j}, ..., \sum_{j=1}^{n} w_j x_{mj}) \quad (1)$$

Эта целевая функция позволяет решить задачу, описание которой приведено в пункте 1.1. Для задач из пунктов 1.2 и 1.3 целевая функция имеет вид

Minimize

$$z = Max(W_1 - \sum_{j=1}^{n} w_j x_{1j}, ..., W_m \sum_{j=1}^{n} w_j x_{mj}) - Min(W_1 - \sum_{j=1}^{n} w_j x_{1j}, ..., W_m - \sum_{j=1}^{n} w_j x_{mj}) \quad (2)$$

Здесь W_i (i=1, ..., m) – максимальный «вес» объектов, который способно вместить i-тое хранилище. Это позволяет сразу учесть ограничения на вместимость хранилищ. Остальные ограничения задаются отдельно.

Для решения задачи оптимального распределения используется алгоритм, показанный на листинге 1 (язык программирования C#). На вход алгоритма подаются два массива:

Упорядоченный по возрастанию или убыванию массив записей, которые содержат сведения об n объектах. Эти записи содержат по крайней мере два поля, среди которых есть идентификатор и вес объекта обязательные. Массив упорядочивается по весу объекта. В случае задачи из пункта 1.1. эти поля содержат ключ класса эквивалентности и вычислительную сложность декартова произведения соответствующих этому ключу классов эквивалентности таблиц-операндов.

Массив коллекций объектов для каждого хранилища. Записи каждой коллекции также содержат идентификатор и вес item. Размерность этого массива *m*, а каждой коллекции не более $\left\lceil \frac{n}{m} \right\rceil$.

Для задачи из пункта 1.3 вместо коллекции хранилищ может использоваться обычный массив.

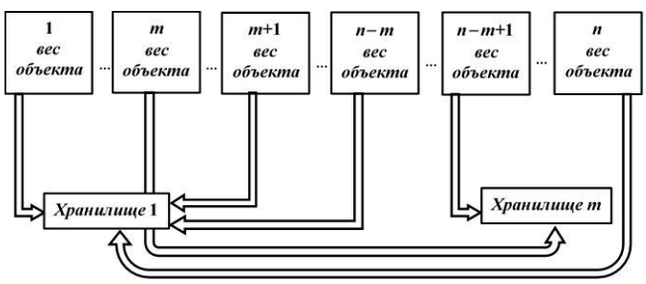
Кроме того, алгоритм получает количество объектов – *n* и хранилищ – *m*.

Listing 1. UDD algorithm

```
public void UDD((<datatype>[] ItemRecords, ArrayList[] Warehou
seRecords, int n, int m)
{
int i, j;
j = 0;
for (i = 0; i < n / 2; i++)
{
WarehouseRecords[j].Add(ItemRecords[i]);
WarehouseRecords[j].Add(ItemRecords[n – 1 – i]);
j = (j + 1) % m;
}
} //End of UDD
```

Приведенный алгоритм относится к классу жадных алгоритмов и распределяет объекты по хранилищам так, как это показано на рисунке 1. На каждом шаге выполнения алгоритма в очередное хранбилище помешаются два объекта, симметричные относительно середины массива (их индексы задаются в строках 4, 5). Индекс хранилища определяется как остаток от деления текущего индекса массива ItemRecords на количество хранилищ. Таким образом, в хранилище помещаются наибольший и наименьший из нераспределенных объектов. Это отличает предложенный алгоритм от большинства подобных алгоритмов.

Замечание. Если n нечетное число, то последняя пара объектов помешается в очередное хранилище после окончания цикла.



Р и с. 1. Распределение объектов по хранилищам

F i g. 1. Distribution of objects on storages

Для задач из пунктов 1.1 и 1.2 заполнение хранилища производится до тех пор, пока выполняется ограничение на объем хранилища.

Вычислительная сложность алгоритма складывается из вычислительной сложности операций сортировки массива объектов и однократного просмотра этого массива, следовательно, имеет порядок $O(n^2+n)$.

3. Анализ алгоритма оптимального распределения

Поскольку предложенный алгоритм относится к классу эвристических жадных алгоритмов, то оценить его качественные характеристики можно только экспериментальным путем. Поэтому для анализа алгоритма оптимального распределения был проведен ряд экспериментов. Цель этих экспериментов состояла в том, чтобы определить зависимости:

- времени выполнения алгоритма от количества объектов;
- качества распределения (разности между максимумом и минимумом заполнения хранилищ) от количества хранилищ и интервала значений весов объектов.

Анализ алгоритма производился для трех типов задач, рассмотренных в разделе 1. Далее приводятся результаты экспериментов для каждой задачи.

Все эксперименты проводились на рабочей станции с такими характеристиками: процессор Intel® Core™ i7-8700K и 32 ГБ оперативной памяти.

3.1. *Симметричное горизонтальное распределение таблиц для операции Join*

В этой серии экспериментов хранилищ изменялось по степеням 2, от 2^3 до 2^{16} (в статье приведены результаты для всех значений). Количество объектов изменялось от $12 \cdot 10^6$ до $24 \cdot 10^7$ с шагом $12 \cdot 10^6$. Веса объектов изменялись в интервалах от w+1 до w+20000 (w=0, 20000, 40000, ..., 100000) и в интервалах от w+1 до w+$2 \cdot 10^6$ (w=0, $2 \cdot 10^6$, $4 \cdot 10^6$,..., $10 \cdot 10^6$). Для количества объектов в статье приведены результаты только для граничных значений, а для весов еще и для одного

промежуточного значения.

В каждом эксперименте производилось распределение объектов, веса которых находились в заданном интервале между заданным числом хранилищ. Затем определялись величины Wmax и Wmin для наиболее и наименее заполненных хранилищ. Затем по формуле

$$\frac{W_{\max} - W_{\min}}{\frac{W_{\max} + W_{\min}}{2}} \times 100 \qquad (3)$$

вычислялась величина (в процентах), характеризующая качество распределения.

Оптимизация осуществлялось на основе целевой функции (1) без ограничений

Далее приводятся результаты экспериментов.

Эксперимент1. В этом эксперименте проводился анализ поведения алгоритма при следующих условиях: количество объектов равно $12 \cdot 10^6$, наибольшее значение веса объекта отличается от наименьшего не более, чем на 20000 (минимум) и $2 \cdot 10^6$ (максимум) единиц. Результат эксперимента приведен в таблице 1.

Таблица 1. Качество распределения для $12 \cdot 10^6$ объектов.

Table 1. Distribution quality for 12×10^6 objects

n	**12000000**					
m	**Интервалы значений объемов классов эквивалентности**					
	1-20000	**60001-80000**	**100001-120000**	**1-2·10⁶Ч**	**6·10⁶-8·10⁶**	**10·10⁶-12·10⁶**
8	0,00000071	0,00000010	0,00000006	0,00000007	0,00000001	0,00000001
16	0,00000181	0,00000017	0,00000018	0,00000010	0,00000002	0,00000002
32	0,00000605	0,00000069	0,00000041	0,00000026	0,00000004	0,00000003
64	0,00001056	0,00000146	0,00000073	0,00000035	0,00000008	0,00000005
128	0,00001632	0,00000274	0,00000111	0,00000116	0,00000016	0,00000006
256	0,00429285	0,00427032	0,00426881	0,00426939	0,00426693	0,00426678
512	0,00858716	0,00853996	0,00853647	0,00853728	0,00853391	0,00853374
1024	0,01715141	0,01707737	0,01707183	0,01707426	0,01706710	0,01706700
2048	0,03426990	0,03415179	0,03414767	0,03414707	0,03413712	0,03413654
4096	0,06847980	0,06830988	0,06829985	0,06830409	0,06828775	0,06828404
8192	0,13685820	0,13656729	0,13654501	0,13656129	0,13652577	0,13652495
16384	0,27337972	0,27292487	0,27291490	0,27292661	0,27286646	0,27285650
32768	0,54579613	0,54509837	0,54504163	0,54515903	0,54498541	0,54497094
65536	1,09419349	1,09310503	1,09302175	1,09326761	1,09295563	1,09292142

Приведенные в таблице результаты эксперимента показывают, что алгоритм распределения дает достаточно хорошие результаты. При таком распределении время выполнения операции Join будет примерно одинаковым на всех процессорах. Время работы алгоритма составило в среднем 1,47 секунды.

Эксперимент 2. В этом эксперименте проводился анализ поведения алгоритма при следующих условиях: количество объектов равно $24 \cdot 10^7$, наибольшее значение веса объекта, как и в эксперименте 1, отличается от наименьшего не более, чем на 20000 (минимум) и $2 \cdot 10^6$ (максимум) единиц. Результат эксперимента приведен в таблице 2.

Таблица 2. Качество распределения для $12 \cdot 10^6$ объектов.

Table 2. Distribution quality for 12×10^6 objects

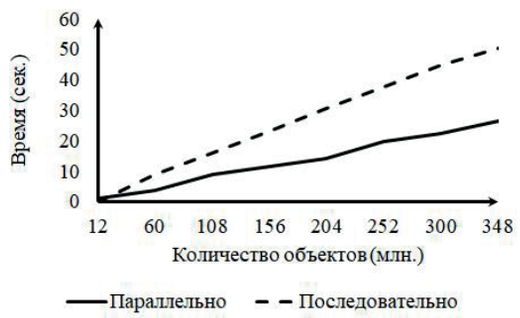
n	**240000000**					
m	**Интервалы значений объемов классов эквивалентности**					
	1-20000	**60001-80000**	**100001-120000**	**1-2·10⁶Ч**	**6·10⁶-8·10⁶**	**10·10⁶-12·10⁶**
8	0,00000005	0,00000001	0,00000001	0,00000001	0,00000001	0,00000000
16	0,00000011	0,00000002	0,00000001	0,00000001	0,00000001	0,00000000
32	0,00000024	0,00000003	0,00000002	0,00000001	0,00000001	0,00000000
64	0,00000039	0,00000005	0,00000003	0,00000005	0,00000001	0,00000000
128	0,00000072	0,00000010	0,00000010	0,00000008	0,00000001	0,00000001
256	0,00000172	0,00000016	0,00000024	0,00000014	0,00000002	0,00000001
512	0,00000378	0,00000034	0,00000037	0,00000019	0,00000003	0,00000001
1024	0,00085758	0,00085390	0,00085385	0,00085362	0,00085338	0,00085336
2048	0,00171473	0,00170758	0,00170751	0,00170721	0,00170674	0,00170672
4096	0,00342762	0,00341466	0,00341445	0,00341414	0,00341347	0,00341346
8192	0,00685273	0,00682862	0,00682956	0,00682784	0,00682679	0,00682677
16384	0,01369133	0,01365629	0,01365698	0,01365509	0,01365303	0,01365300
32768	0,02736578	0,02730840	0,02730973	0,02730736	0,02730416	0,02730405
65536	0,05468587	0,05460827	0,05460853	0,05460651	0,05460057	0,05460054

Результаты этого эксперимента подтверждают выводы, сделанные на основе результатов предыдущего эксперимента. Более того, прослеживается тенденция улучшения качества распределения с увеличением количества объектов. Время работы алгоритма в этом эксперименте составило в среднем 33,1 секунды.

Эксперимент 3. В этом эксперименте было исследована возможность параллельной реализации алгоритма распределения. Был использован параллельный алгоритм быстрой сортировки. Распараллеливание было проведено на двенадцати потоках (двенадцать виртуальных ядер процессора). Результаты распараллеливания приведены в таблице 3 и на рисунке 2.

Таблица 3. Результаты распараллеливания
Table 3. The parallelization results

n	$T_{последовательно}$	$T_{параллельно}$
12Ч	1,47	0,82
60Ч	8,70	3,63
108Ч	15,94	8,69
156Ч	23,17	11,37
204Ч	30,40	14,22
252Ч	37,63	19,52
300Ч	44,87	22,18
348Ч	50,19	26,40



Р и с. 2. Две реализации алгоритма распределения
F i g. 2. Two implementations of the distribution algorithm

Эксперимент показал, что распараллеливание несущественно, всего лишь в два раза, улучшает временные характеристики алгоритма. Учитывая малое время последовательного выполнения алгоритма, можно сделать вывод о том, что в этом случае распараллеливание нецелесообразно.

Результаты обоих экспериментов показали, что предложенный алгоритм имеет хорошую скорость вычислений и дает достаточно качественное распределение объектов по хранилищам. Рассмотренные объемы данных, распределение которых осуществлялось, несомненно относятся к классу big data. Поэтому возможность выполнить распределение данных за малое время и с достаточно хорошей балансировкой объёмов данных (различие составляет не более 2%) способно обеспечить эффективное использование параллельных вычислительных систем (машин баз данных).

3.2. *Логистическое распределение*

3.2.1. *Складская логистика*

Для этого класса задач была использована целевая функция (2). В ней уже учтено ограничение на максимальный размер хранилища. Этого ограничения достаточно для задачи складской логистики, а в задаче распределения бюджетных мест потребуются дополнительные ограничения.

В этом эксперименте проводился анализ поведения алгоритма при следующих условиях: количество объектов равно 50000, наибольшее значение веса объекта, отличается от наименьшего не более, чем на 80 единиц. Результат эксперимента приведен в таблице 4.

Таблица 4. Качество распределения объектов по хранилищам с ограниченным объемом
Table 4. Objects distribution quality in limited volume storage facilities

n	**50000**						
m	8	16	32	64	128	256	512
21-100	0,00240162	0,06724555	0,13551654	0,2818766	0,56727304	1,11177628	2,16606498
101-180	0,00091419	0,06650752	0,13297356	0,2633697	0,51695635	1,05482914	2,09844273

Эксперимент показал, что наличие ограничений на объемы хранилищ несущественно сказывается на качестве распределения.

3.2.2. *Образовательная логистика*

Для каждого вуза количество мест, по которому он имеет возможность подготовки специалистов, учтено в целевой функции (2). Кроме того, в этом случае вводятся дополнительные ограничения, которые отражают потребности регионов по подготовке специалистов, необходимых для регионального развития. Пусть W_R – потребность региона R в специалистах некоторого направления, n_R – количество вузов в этом регионе, осуществляющих подготовку по этому направлению. Это ограничение задается неравенством

$$\sum_{i=1}^{n_R} W_{\alpha_i} \geq W_R$$

(W_{α_i} – номер регионального вуза в общем списке). Для проведения эксперимента были выбраны две специальности: «01.03.02 – Прикладная математика и информатика» и «09.03.03 – Прикладная информатика (по отраслям)». Сведения о количестве вузов, имеющих направления подготовки по этим специальностям взяты на электронном ресурсе[1]. В результате эксперимента были получены результаты, отраженные в таблице 5.

Таблица 5. Качество распределения бюджетных мест с дополнительными ограничениями
Table 5. Distribution quality of budget places with additional restrictions

Специальность	**Количество вузов**	**Количество бюджетных мест**	**Недополучено мест (%)**
01.03.02	**158**	2867	3,17460317
09.03.03	**468**	2590	3,38983051

Выводы

На основе приведенных в статье описания эвристического алгоритма оптимального распределения и результатов проведенных экспериментов позволяют сделать следующие выводы.

Алгоритм достаточно прост и может быть легко реализован в любом языке программирования. Авторы реализовали его, в том числе, на языке Transact-Sql. Выбор языка программирования определяется конкретными условиями задачи распределения предметов.

Время работы алгоритма для big data мало, что позволяет эффективно использовать его при подготовке данных для параллельного решения задач с высокой вычислительной сложностью.

[1] Специальности бакалавриата и специалитета [Электронный ресурс]. URL: http://vuzoteka.ru/вузы/специальности (дата обращения 20.07.2019).

Несмотря на то, что алгоритм эвристический, он показывает хорошие результаты при распределении объектов по хранилищам. Наибольшее, в смысле удовлетворения целевой функции, заполнение хранилища отличается от наименьшего на незначительную величину.

Приведенные в статье результаты соответствуют начальной стадии исследования алгоритма. Дальнейшие исследования авторы предполагают направить на изучение его свойств при наличии сложных систем ограничений на объемы хранилищ и для различных целевых функций.

References

[1] Munerman V., Munerman D. Realization of Distributed Data Processing on the Basis of Container Technology. In: *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus),* Saint Petersburg and Moscow, Russia, 2019, pp. 1740-1744. (In Eng.) DOI: 10.1109/EIConRus.2019.8656766

[2] Zakharov V., Kirikova A., Munerman V., Samoilova T. Architecture of Software-Hardware Complex for Searching Images in Database. In: *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Saint Petersburg and Moscow, Russia, 2019, pp. 1735-1739. (In Eng.) DOI: 10.1109/EIConRus.2019.8657241

[3] Mohiuddin I. et al. Secure distributed adaptive bin packing algorithm for cloud storage. *Future Generation Computer Systems.* 2019; 90:307-316. (In Eng.) DOI: 10.1016/j.future.2018.08.013

[4] Akbar M.M., Manning E.G., Shoja G.C., Khan S. Heuristic Solutions for the Multiple-Choice Multi-dimension Knapsack Problem. In: Alexandrov V.N., Dongarra J.J., Juliano B.A., Renner R.S., Tan C.J.K. (eds) *Computational Science - ICCS 2001. ICCS 2001. Lecture Notes in Computer Science,* vol. 2074. Springer, Berlin, Heidelberg, 2001, pp. 659-668. (In Eng.) DOI: 10.1007/3-540-45718-6_71

[5] Fréville A. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research.* 2004; 155(1):1-21. (In Eng.) DOI: 10.1016/S0377-2217(03)00274-1

[6] Trivella A., Pisinger D. The load-balanced multi-dimensional bin-packing problem. *Computers & Operations Research.* 2016; 74:152-164. (In Eng.) DOI: 10.1016/j.cor.2016.04.020

[7] Delorme M., Iori M. Enhanced Pseudo-polynomial Formulations for Bin Packing and Cutting Stock Problems. *INFORMS Journal on Computing*. 2019. (In Eng.) DOI: 10.1287/ijoc.2018.0880

[8] Abdel-Basset M., Manogaran G., Abdel-Fatah L. et al. An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems. *Personal and Ubiquitous Computing*. 2018; 22(5-6):1117-1132. (In Eng.) DOI: 10.1007/s00779-018-1132-7

[9] Li H., Zhu G., Cui C., Tang H., Dou Y., He C. Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing. *Computing.* 2016; 98(3):303-317. (In Eng.) DOI: 10.1007/s00607-015-0467-4

[10] Usmani Z., Singh S. A Survey of Virtual Machine Placement Techniques in a Cloud Data Center. *Procedia Computer Science*. 2016; 78:491-498. (In Eng.) DOI: 10.1016/j.procs.2016.02.093

[11] Sridhar R., Chandrasekaran M., Page T. Optimization of heterogeneous Bin packing using adaptive genetic algorithm. *IOP Conference Series: Materials Science and Engineering.* 2017; 183(1):012026. (In Eng.) DOI: 10.1088/1757-899X/183/1/012026

[12] Quiroz-Castellanos M., Cruz-Reyes L., Torres-Jimenez J., Gómez C.S., Fraire Huacujaa H.J., Alvim A.C.F. A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research.* 2015; 55:52-64. (In Eng.) DOI: 10.1016/j.cor.2014.10.010

Поступила 20.07.2019; принята к публикации 10.08.2019; опубликована онлайн 30.09.2019.
Submitted 20.07.2019; revised 10.08.2019; published online 30.09.2019.

Об авторах:

Мунерман Виктор Иосифович, доцент кафедры информатики, Смоленский государственный университет (214000, Россия, г. Смоленск, ул. Пржевальского, д. 4), кандидат технических наук, ORCID: http://orcid.org/0000-0002-9628-4049, vimoon@gmail.com

Мунерман Даниил Викторович, лаборант-стажер кафедры информатики, Смоленский государственный университет (214000, Россия, г. Смоленск, ул. Пржевальского, д. 4), ORCID: http://orcid.org/0000-0002-5139-6645, danvmoon@gmail.com

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Victor I. Munerman, Associate Professor of the Department of Informatics, Smolensk State University (4 Przhevalsky Str., Smolensk 4214000, Russia), Ph.D. (Engineering), ORCID: http://orcid.org/0000-0002-9628-4049, vimoon@gmail.com

Daniel V. Munerman, Laboratory Assistant of the Department of Informatics, Smolensk State University (4 Przhevalsky Str., Smolensk 4214000, Russia), ORCID: http://orcid.org/0000-0002-5139-6645, danvmoon@gmail.com

All authors have read and approved the final manuscript.